OOPS Interview questions

What is the full form of OOPS?

Object Oriented Programming System.

What is a class?

Class is a blue print which reflects the entities attributes and actions. Technically defining a class is designing an user defined data type.

What is an object?

An instance of the class is called as object.

List the types of inheritance supported in C++.

Single, Multilevel, Multiple, Hierarchical and Hybrid.

What is the role of protected access specifier?

If a class member is protected then it is accessible in the inherited class. However, outside the both the private and protected members are not accessible.

What is encapsulation?

The process of binding the data and the functions acting on the data together in an entity (class) called as encapsulation.

What is abstraction?

Abstraction refers to hiding the internal implementation and exhibiting only the necessary details.

What is inheritance?

Inheritance is the process of acquiring the properties of the exiting class into the new class. The existing class is called as base/parent class and the inherited class is called as derived/child class.

Explain the purpose of the keyword volatile.

Declaring a variable volatile directs the compiler that the variable can be changed externally. Hence avoiding compiler optimization on the variable reference.

What is an inline function?

A function prefixed with the keyword inline before the function definition is called as inline function. The inline functions are faster in execution when compared to normal functions as the compiler treats inline functions as macros.

What is a storage class?

Storage class specifies the life or scope of symbols such as variable or functions.

Mention the storage classes names in C++.

The following are storage classes supported in C++

auto, static, extern, register and mutable

What is the role of mutable storage class specifier?

A constant class object's member variable can be altered by declaring it using mutable storage class specifier. Applicable only for non-static and non-constant member variable of the class.

Distinguish between shallow copy and deep copy.

Shallow copy does memory dumping bit-by-bit from one object to another. Deep copy is copy field by field from object to another. Deep copy is achieved using copy constructor and or overloading assignment operator.

What is a pure virtual function?

A virtual function with no function body and assigned with a value zero is called as pure virtual function.

What is an abstract class in C++?

A class with at least one pure virtual function is called as abstract class. We cannot instantiate an abstract class.

What is a reference variable in C++?

A reference variable is an alias name for the existing variable. Which mean both the variable name and reference variable point to the same memory location. Therefore updation on the original variable can be achieved using reference variable too.

What is role of static keyword on class member variable?

A static variable does exit though the objects for the respective class are not created. Static member variable share a common memory across all the objects created for the respective class. A static member variable can be referred using the class name itself.

Explain the static member function.

A static member function can be invoked using the class name as it exits before class objects comes into existence. It can access only static members of the class.

Name the data type which can be used to store wide characters in C++.

wchar_t

What are/is the operator/operators used to access the class members?

Dot (.) and Arrow (->)

Can we initialize a class/structure member variable as soon as the same is defined?

No, Defining a class/structure is just a type definition and will not allocated memory for the same.

What is the data type to store the Boolean value?

bool, is the new primitive data type introduced in C++ language.

What is function overloading?

Defining several functions with the same name with unique list of parameters is called as function overloading.

What is operator overloading?

Defining a new job for the existing operator w.r.t the class objects is called as operator overloading.

Do we have a String primitive data type in C++?

No, it’s a class from STL (Standard template library).

Name the default standard streams in C++.

cin, cout, cerr and clog.

Which access specifier/s can help to achive data hiding in C++?

Private & Protected.

When a class member is defined outside the class, which operator can be used to associate the function definition to a particular class?

Scope resolution operator (::)

What is a destructor? Can it be overloaded?

A destructor is the member function of the class which is having the same name as the class name and prefixed with tilde (~) symbol. It gets executed automatically w.r.t the object as soon as the object loses its scope. It cannot be overloaded and the only form is without the parameters.

What is a constructor?

A constructor is the member function of the class which is having the same as the class name and gets executed automatically as soon as the object for the respective class is created.

What is a default constructor? Can we provide one for our class?

Every class does have a constructor provided by the compiler if the programmer doesn't provides one and known as default constructor. A programmer provided constructor with no parameters is called as default constructor. In such case compiler doesn't provides the constructor.

Which operator can be used in C++ to allocate dynamic memory?

'new' is the operator can be used for the same.

What is the purpose of 'delete' operator?

'delete' operator is used to release the dynamic memory which was created using 'new' operator.

Can I use malloc() function of C language to allocate dynamic memory in C++?

Yes, as C is the subset of C++, we can all the functions of C in C++ too.

Can I use 'delete' operator to release the memory which was allocated using malloc() function of C language?

No, we need to use free() of C language for the same.

What is a friend function?

A function which is not a member of the class but still can access all the member of the class is called so. To make it happen we need to declare within the required class following the keyword ‘friend’.

What is a copy constructor?

A copy constructor is the constructor which take same class object reference as the parameter. It gets automatically invoked as soon as the object is initialized with another object of the same class at the time of its creation.

Does C++ supports exception handling? If so what are the keywords involved in achieving the same.

C++ does supports exception handling. try, catch & throw are keyword used for the same.

Explain the pointer – this.

This, is the pointer variable of the compiler which always holds the current active object’s address.

What is the difference between the keywords struct and class in C++?

By default the members of struct are public and by default the members of the class are private.

Can we implement all the concepts of OOPS using the keyword struct?

Yes.

What is the block scope variable in C++?

A variable whose scope is applicable only within a block is said so. Also a variable in C++ can be declared anywhere within the block.

What is the role of the file opening mode ios::trunk?

If the file already exists, its content will be truncated before opening the file.

What is the scope resolution operator?

The scope resolution operator is used to

Resolve the scope of global variables.

To associate function definition to a class if the function is defined outside the class.

What is a namespace?

A namespace is the logical division of the code which can be used to resolve the name conflict of the identifiers by placing them under different name space.

What are command line arguments?

The arguments/parameters which are sent to the main() function while executing from the command line/console are called so. All the arguments sent are the strings only.

What is a class template?

A template class is a generic class. The keyword template can be used to define a class template.

How can we catch all kind of exceptions in a single catch block?

The catch block with ellipses as follows

```
catch(…)

{

}
```

What is keyword auto for?

By default every local variable of the function is automatic (auto). In the below function both the variables ‘i’ and ‘j’ are automatic variables.

```
void f()

{

   int i;


   auto int j;

}
```

NOTE − A global variable can’t be an automatic variable.

What is a static variable?

A static local variables retains its value between the function call and the default value is 0. The following function will print 1 2 3 if called thrice.

```
void f()
{
   static int i;


   ++i;
   printf("%d ",i);
}
```

If a global variable is static then its visibility is limited to the same source code.

What is the purpose of extern storage specifier.

Used to resolve the scope of global symbol

```
#include <iostream>
using namespace std;
main() {
   extern int i;


   cout<<i<<endl;
}
int i = 20;
```

What is the meaning of base address of the array?

The starting address of the array is called as the base address of the array.

When should we use the register storage specifier?

If a variable is used most frequently then it should be declared using register storage specifier, then possibly the compiler gives CPU register for its storage to speed up the look up of the variable.

Can a program be compiled without main() function?

Yes, it can be but cannot be executed, as the execution requires main() function definition.

Where an automatic variable is stored?

Every local variable by default being an auto variable is stored in stack memory

What is a container class?

A class containing at least one member variable of another class type in it is called so.

What is a token?

A C++ program consists of various tokens and a token is either a keyword, an identifier, a constant, a string literal, or a symbol.

What is a preprocessor?

Preprocessor is a directive to the compiler to perform certain things before the actual compilation process begins.

What are the different ways of passing parameters to the functions? Which to use when?

Call by value − We send only values to the function as parameters. We choose this if we do not want the actual parameters to be modified with formal parameters but just used.

Call by address − We send address of the actual parameters instead of values. We choose this if we do want the actual parameters to be modified with formal parameters.

Call by reference − The actual parameters are received with the C++ new reference variables as formal parameters. We choose this if we do want the actual parameters to be modified with formal parameters.

What is reminder for 5.0 % 2?

Error, It is invalid that either of the operands for the modulus operator (%) is a real number.

Which compiler switch to be used for compiling the programs using math library with g++ compiler?

Opiton –lm to be used as > g++ –lm <file.cpp>

Can we resize the allocated memory which was allocated using ‘new’ operator?

No, there is no such provision available.

Which operator can be used to determine the size of a data type/class or variable/object?

sizeof

How can we refer to the global variable if the local and the global variable names are same?

We can apply scope resolution operator (::) to the for the scope of global variable.

What are valid operations on pointers?

The only two permitted operations on pointers are

Comparision ii) Addition/Substraction (excluding void pointers)

What is recursion?

Function calling itself is called as recursion.

What is the first string in the argument vector w.r.t command line arguments?

Program name.

What is the maximum length of an identifier?

Ideally it is 32 characters and also implementation dependent.

What is the default function call method?

By default the functions are called by value.

What are available mode of inheritance to inherit one class from another?

Public, private & protected

What is the difference between delete and delete[]?

Delete[] is used to release the array allocated memory which was allocated using new[] and delete is used to release one chunk of memory which was allocated using new.

Does an abstract class in C++ need to hold all pure virtual functions?

Not necessarily, a class having at least one pure virtual function is abstract class too.

Is it legal to assign a base class object to a derived class pointer?

No, it will be error as the compiler fails to do conversion.

What happens if an exception is thrown outside a try block?

The program shall quit abruptly.

Are the exceptions and error same?

No, exceptions can be handled whereas program cannot resolve errors.

What is function overriding?

Defining the functions within the base and derived class with the same signature and name where the base class's function is virtual.

Which function is used to move the stream pointer for the purpose of reading data from stream?

seekg()

Which function is used to move the stream pointer for the purpose of writing data from stream?

seekp()

Are class functions taken into consideration as part of the object size?

No, only the class member variables determines the size of the respective class object.

Can we create and empty class? If so what would be the size of such object.

We can create an empty class and the object size will be 1.

What is 'std'?

Default namespace defined by C++.

What is the full form of STL?

Standard template library

What is 'cout'?

cout is the object of ostream class. The stream 'cout' is by default connected to console output device.

What is 'cin'?

cin is the object of istream class. The stream 'cin' is by default connected to console input device.

What is the use of the keyword 'using'?

It is used to specify the namespace being used in.

If a pointer declared for a class, which operator can be used to access its class members?

Arrow (->) operator can be used for the same

What is difference between including the header file with-in angular braces < > and double quotes " "

If a header file is included with in < > then the compiler searches for the particular header file only with in the built in include path. If a header file is included with in " ", then the compiler searches for the particular header file first in the current working directory, if not found then in the built in include path

S++ or S = S+1, which can be recommended to increment the value by 1 and why?

S++, as it is single machine instruction (INC) internally.

What is the difference between actual and formal parameters?

The parameters sent to the function at calling end are called as actual parameters while at the receiving of the function definition called as formal parameters.

What is the difference between variable declaration and variable definition?

Declaration associates type to the variable whereas definition gives the value to the variable.

Which key word is used to perform unconditional branching?

goto.

Is 068 a valid octal number?

No, it contains invalid octal digits.

What is the purpose of #undef preprocessor?

It will be used to undefine an existing macro definition

Can we nest multi line comments in a C++ code?

No, we cannot.

What is a virtual destructor?

A virtual destructor ensures that the objects resources are released in the reverse order of the object being constructed w.r.t inherited object.

What is the order of objects destroyed in the memory?

The objects are destroyed in the reverse order of their creation.

What is a friend class?

A class members can gain accessibility over other class member by placing the class declaration prefixed with the keyword 'friend' in the destination class.

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

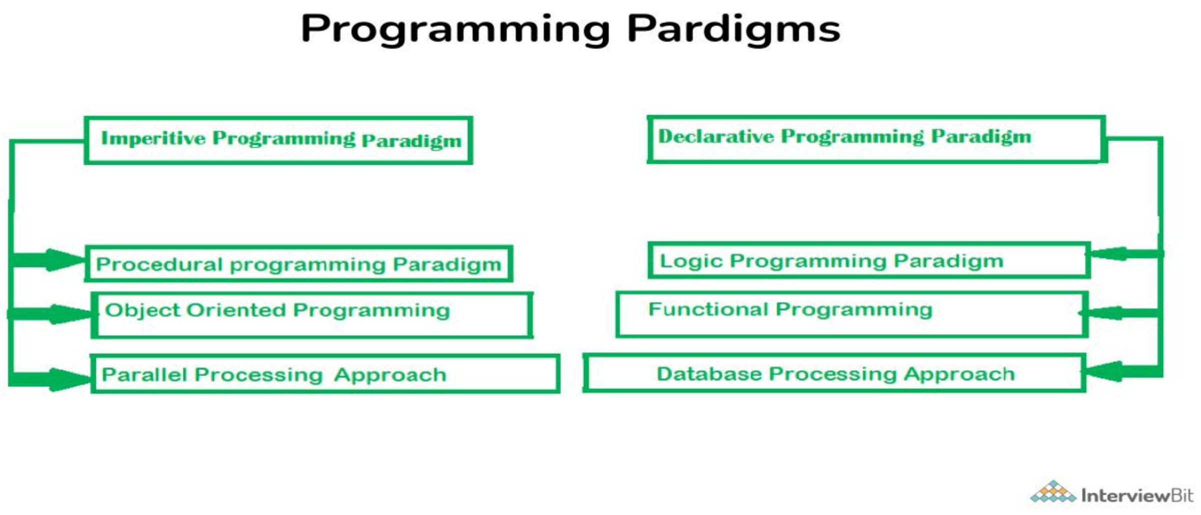
Programming paradigms refers to the method of classification of programming languages based on their features. There are mainly two types of Programming Paradigms:

- Imperative Programming Paradigm
- Declarative Programming Paradigm

Now, these paradigms can be further classified based:

1. Imperative Programming Paradigm: Imperative programming focuses on HOW to execute program logic and defines control flow as statements that change a program state. This can be further classified as:
a) Procedural Programming Paradigm: Procedural programming specifies the steps a program must take to reach the desired state, usually read in order from top to bottom.
b) Object-Oriented Programming or OOP: Object-oriented programming (OOP) organizes programs as objects, that contain some data and have some behavior.
c) Parallel Programming: Parallel programming paradigm breaks a task into subtasks and focuses on executing them simultaneously at the same time.

2. Declarative Programming Paradigm: Declarative programming focuses on WHAT to execute and defines program logic, but not a detailed control flow. Declarative paradigm can be further classified into:
a) Logical Programming Paradigm: Logical programming paradigm is based on formal logic, which refers to a set of sentences expressing facts and rules about how to solve a problem
b) Functional Programming Paradigm: Functional programming is a programming paradigm where programs are constructed by applying and composing functions.
c) Database Programming Paradigm: Database programming model is used to manage data and information structured as fields, records, and files.


Programming Pardigms
Imperitive Programming Paradigm
Procedural programming Paradigm
Object Oriented Programming
Parallel Processing Approach
Declarative Programming Paradigm
Logic Programming Paradigm
Functional Programming
Database Processing Approach
InterviewBit

5. What is meant by Structured Programming?

Structured Programming refers to the method of programming which consists of a completely structured control flow. Here structure refers to a block, which contains a set of rules, and has a definitive control flow, such as (if/then/else), (while and for), block structures, and subroutines.

Nearly all programming paradigms include Structured programming, including the OOPs model.

6. What are the main features of OOPs?

OOPs or Object Oriented Programming mainly comprises of the below four features, and make sure you don't miss any of these:

- Inheritance
- Encapsulation
- Polymorphism
- Data Abstraction

7. What are some advantages of using OOPs?

- OOPs is very helpful in solving very complex level of problems.
- Highly complex programs can be created, handled, and maintained easily using object-oriented programming.
- OOPs, promote code reuse, thereby reducing redundancy.
- OOPs also helps to hide the unnecessary details with the help of Data Abstraction.
- OOPs, are based on a bottom-up approach, unlike the Structural programming paradigm, which uses a top-down approach.
- Polymorphism offers a lot of flexibility in OOPs.

8. Why is OOPs so popular?

OOPs programming paradigm is considered as a better style of programming. Not only it helps in writing a complex piece of code easily, but it also allows users to handle and maintain them easily as well. Not only that, the main pillar of OOPs - Data Abstraction, Encapsulation, Inheritance, and Polymorphism, makes it easy for programmers to solve complex scenarios. As a result of these, OOPs is so popular.

9. What is a class?

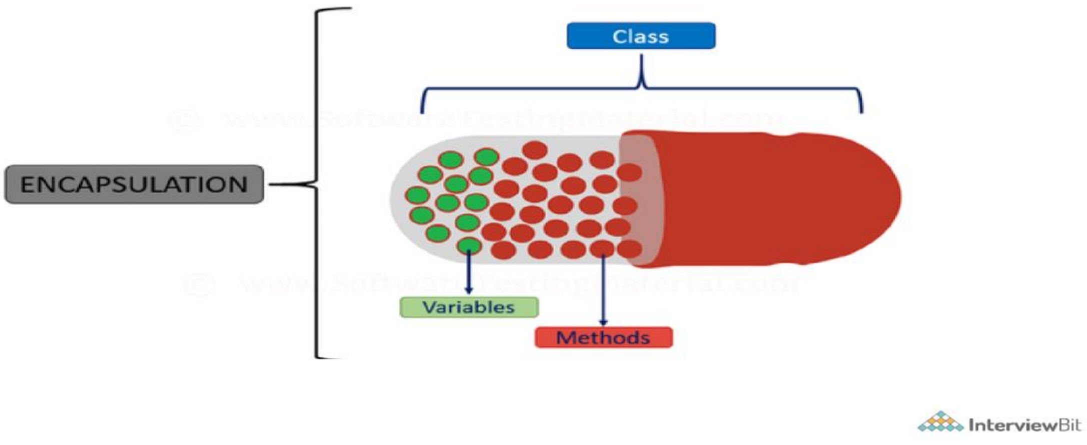
A class can be understood as a template or a blueprint, which contains some values, known as member data or member, and some set of rules, known as behaviors or functions. So when an object is created, it automatically takes the data and functions that are defined in the class.
Therefore the class is basically a template or blueprint for objects. Also one can create as many objects as they want based on a class.

For example, first, a car’s template is created. Then multiple units of car are created based on that template.

10. What is an object?

An object refers to the instance of the class, which contains the instance of the members and behaviors defined in the class template. In the real world, an object is an actual entity to which a user interacts, whereas class is just the blueprint for that object. So the objects consume space and have some characteristic behavior. For example, a specific car.

11. What is encapsulation?



One can visualize Encapsulation as the method of putting everything that is required to do the job, inside a capsule and presenting that capsule to the user. What it means is that by Encapsulation, all the necessary data and methods are bind together and all the unnecessary details are hidden to the normal user. So Encapsulation is the process of binding data members and methods of a program together to do a specific job, without revealing unnecessary details.

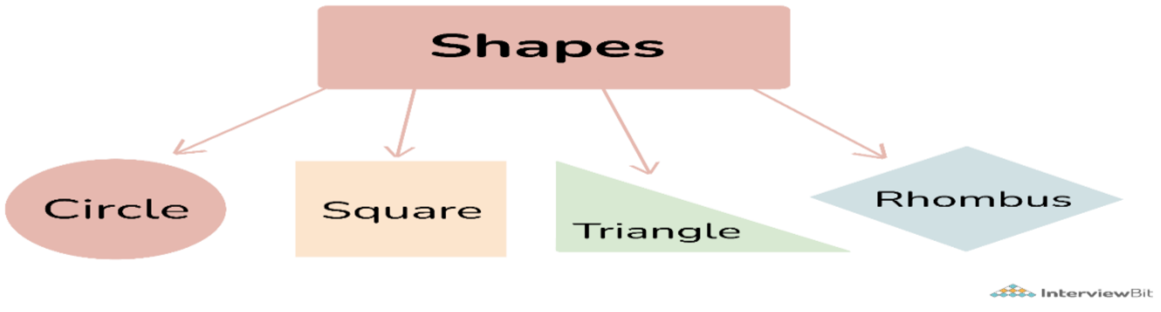
Encapsulation can also be defined in two different ways:

1) **Data hiding:** Encapsulation is the process of hiding unwanted information, such as restricting access to any member of an object.

2) **Data binding:** Encapsulation is the process of binding the data members and the methods together as a whole, as a class.

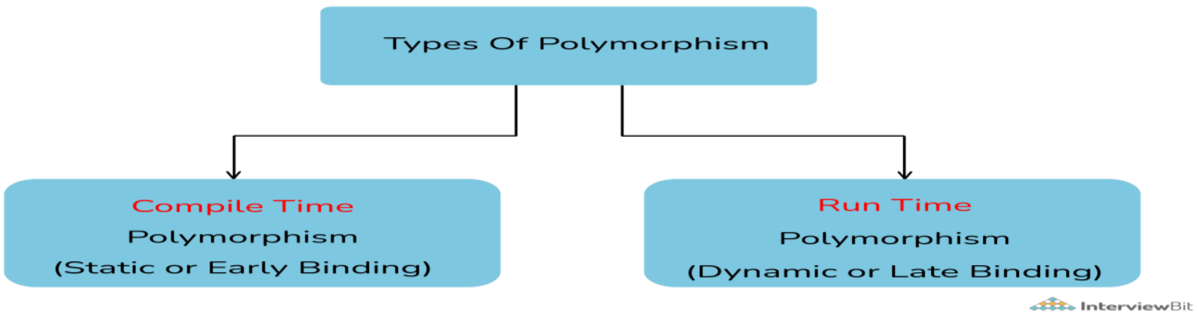
12. What is Polymorphism?

Polymorphism is composed of two words - “poly” which means “many”, and “morph” which means “shapes”. Therefore Polymorphism refers to something that has many shapes.



In OOPs, Polymorphism refers to the process by which some code, data, method, or object behaves differently under different circumstances or contexts. Compile-time polymorphism and Run time polymorphism are the two types of polymorphisms in OOPs languages.

13. What is Compile time Polymorphism and how is it different from Runtime Polymorphism?


Types Of Polymorphism
Compile Time Polymorphism (Static or Early Binding)
Run Time Polymorphism (Dynamic or Late Binding)
InterviewBit

Compile Time Polymorphism: Compile time polymorphism, also known as Static Polymorphism, refers to the type of Polymorphism that happens at compile time. What it means is that the compiler decides what shape or value has to be taken by the entity in the picture.

Example:

```
// In this program, we will see how multiple functions are created with the same name,
// but the compiler decides which function to call easily at the compile time itself.
class CompileTimePolymorphism{
  // 1st method with name add
  public int add(int x, int y){
  return x+y;
  }
  // 2nd method with name add
  public int add(int x, int y, int z){
  return x+y+z;
  }
  // 3rd method with name add
  public int add(double x, int y){
  return (int)x+y;
  }
  // 4th method with name add
  public int add(int x, double y){
  return x+(int)y;
  }
}
class Test{
  public static void main(String[] args){
```

```
  CompileTimePolymorphism demo=new CompileTimePolymorphism();
  // In the below statement, the Compiler looks at the argument types and decides to
call method 1
  System.out.println(demo.add(2,3));
  // Similarly, in the below statement, the compiler calls method 2
  System.out.println(demo.add(2,3,4));
  // Similarly, in the below statement, the compiler calls method 4
  System.out.println(demo.add(2,3.4));
  // Similarly, in the below statement, the compiler calls method 3
  System.out.println(demo.add(2.5,3));
  }
}
```

In the above example, there are four versions of add methods. The first method takes two parameters while the second one takes three. For the third and fourth methods, there is a change of order of parameters. The compiler looks at the method signature and decides which method to invoke for a particular method call at compile time.

Runtime Polymorphism: Runtime polymorphism, also known as Dynamic Polymorphism, refers to the type of Polymorphism that happens at the run time. What it means is it can't be decided by the compiler. Therefore what shape or value has to be taken depends upon the execution. Hence the name Runtime Polymorphism.

Example:

```
class AnyVehicle{
  public void move(){
  System.out.println("Any vehicle should move!!");
  }
}
class Bike extends AnyVehicle{
  public void move(){
  System.out.println("Bike can move too!!");
  }
}
class Test{
  public static void main(String[] args){
  AnyVehicle vehicle = new Bike();
  // In the above statement, as you can see, the object vehicle is of type AnyVehicle
  // But the output of the below statement will be "Bike can move too!!",
  // because the actual implementation of object 'vehicle' is decided during runtime
vehicle.move();
  vehicle = new AnyVehicle();
  // Now, the output of the below statement will be "Any vehicle should move!!",
  vehicle.move();
  }
}
```

As the method to call is determined at runtime, as shown in the above code, this is called runtime polymorphism.

14. How does C++ support Polymorphism?

C++ is an Object-oriented programming language and it supports Polymorphism as well:

- Compile Time Polymorphism: C++ supports compile-time polymorphism with the help of features like templates, function overloading, and default arguments.
- Runtime Polymorphism: C++ supports Runtime polymorphism with the help of features like virtual functions. Virtual functions take the shape of the functions based on the type of object in reference and are resolved at runtime.

15. What is meant by Inheritance?

The term "inheritance" means "receiving some quality or behavior from a parent to an offspring." In object-oriented programming, inheritance is the mechanism by which an object or class (referred to as a child) is created using the definition of another object or class (referred to as a parent). Inheritance not only helps to keep the implementation simpler but also helps to facilitate code reuse.

16. What is Abstraction?

If you are a user, and you have a problem statement, you don't want to know how the components of the software work, or how it's made. You only want to know how the software solves your problem. Abstraction is the method of hiding unnecessary details from the necessary ones. It is one of the main features of OOPs.
For example, consider a car. You only need to know how to run a car, and not how the wires are connected inside it. This is obtained using Abstraction.

17. How much memory does a class occupy?

Classes do not consume any memory. They are just a blueprint based on which objects are created. Now when objects are created, they actually initialize the class members and methods and therefore consume memory.

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

No. An object is necessary to be created if the base class has non-static methods. But if the class has static methods, then objects don't need to be created. You can call the class method directly in this case, using the class name.

19. What is a constructor?

Constructors are special methods whose name is the same as the class name. The constructors serve the special purpose of initializing the objects.
For example, suppose there is a class with the name “MyClass”, then when you instantiate this class, you pass the syntax:
MyClass myClassObject = new MyClass();

Now here, the method called after “new” keyword - MyClass(), is the constructor of this class. This will help to instantiate the member data and methods and assign them to the object myClassObject.

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

The most common classification of constructors includes:

Default constructor: The default constructor is the constructor which doesn’t take any argument. It has no parameters.

```
class ABC
{
  int x;

  ABC()
  {
     x = 0;
  }
}
```

Parameterized constructor: The constructors that take some arguments are known as parameterized constructors.

```
class ABC
{
  int x;

  ABC(int y)
  {
     x = y;
  }
}
```

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

```
class ABC
{
  int x;

  ABC(int y)
  {
     x = y;
  }
  // Copy constructor
  ABC(ABC abc)
  {
     x = abc.x;
  }
}
```

21. What is a copy constructor?

Copy Constructor is a type of constructor, whose purpose is to copy an object to another. What it means is that a copy constructor will clone an object and its values, into another object, is provided that both the objects are of the same class.

22. What is a destructor?

Contrary to constructors, which initialize objects and specify space for them, Destructors are also special methods. But destructors free up the resources and memory occupied by an object. Destructors are automatically called when an object is being destroyed.

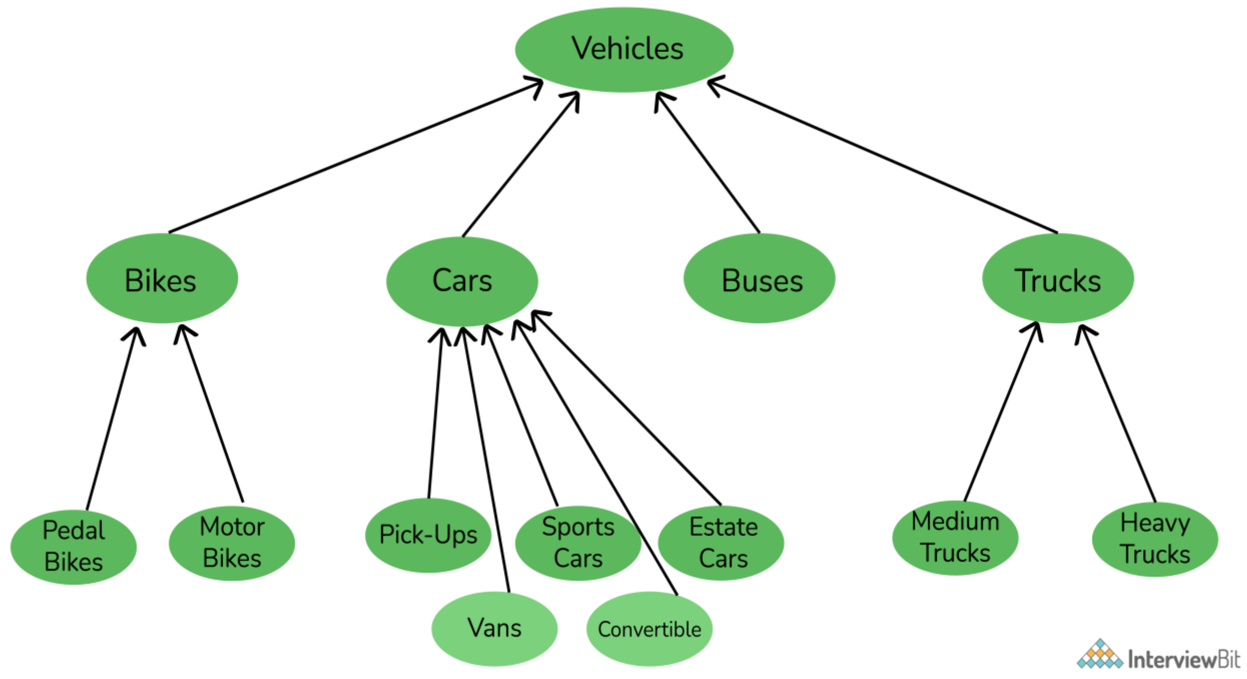
23. Are class and structure the same? If not, what's the difference between a class and a structure?

No, class and structure are not the same. Though they appear to be similar, they have differences that make them apart. For example, the structure is saved in the stack memory, whereas the class is saved in the heap memory. Also, Data Abstraction cannot be achieved with the help of structure, but with class, Abstraction is majorly used.

24. Explain Inheritance with an example?

Inheritance is one of the major features of object-oriented programming, by which an entity inherits some characteristics and behaviors of some other entity and makes them their own. Inheritance helps to improve and facilitate code reuse.

Let me explain to you with a common example. Let's take three different vehicles - a car, truck, or bus. These three are entirely different from one another with their own specific characteristics and behavior. But. in all three, you will find some common elements, like steering wheel, accelerator, clutch, brakes, etc. Though these elements are used in different vehicles, still they have their own features which are common among all vehicles. This is achieved with inheritance. The car, the truck, and the bus have all inherited the features like steering wheel, accelerator, clutch, brakes, etc, and used them as their own. Due to this, they did not have to create these components from scratch, thereby facilitating code reuse.



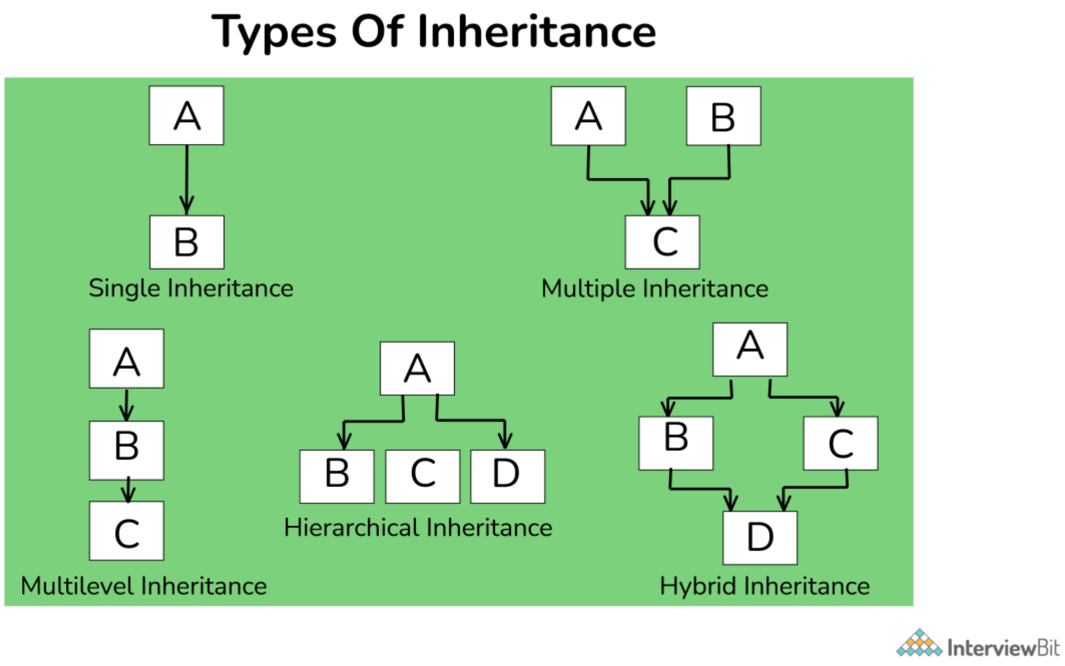
25. Are there any limitations of Inheritance?

Yes, with more powers comes more complications. Inheritance is a very powerful feature in OOPs, but it has some limitations too. Inheritance needs more time to process, as it needs to navigate through multiple classes for its implementation. Also, the classes involved in Inheritance - the base class and the child class, are very tightly coupled together. So if one needs to make some changes, they might need to do nested changes in both classes. Inheritance might be complex for implementation, as well. So if not correctly implemented, this might lead to unexpected errors or incorrect outputs.

26. What are the various types of inheritance?

The various types of inheritance include:

- Single inheritance
- Multiple inheritances
- Multi-level inheritance
- Hierarchical inheritance
- Hybrid inheritance

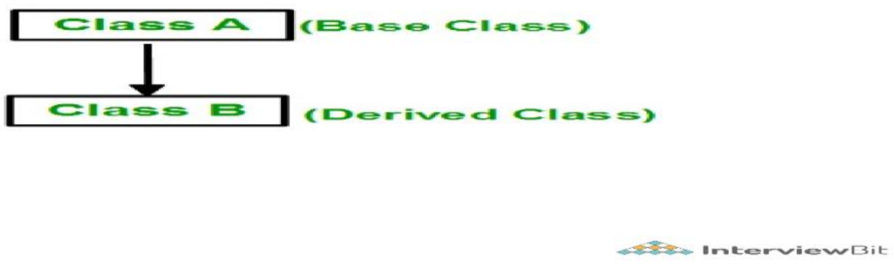


27. What is a subclass?

The subclass is a part of Inheritance. The subclass is an entity, which inherits from another class. It is also known as the child class.

28. Define a superclass?

Superclass is also a part of Inheritance. The superclass is an entity, which allows subclasses or child classes to inherit from itself.



29. What is an interface?

An interface refers to a special type of class, which contains methods, but not their definition. Only the declaration of methods is allowed inside an interface. To use an interface, you cannot create objects. Instead, you need to implement that interface and define the methods for their implementation.

30. What is meant by static polymorphism?

Static Polymorphism is commonly known as the Compile time polymorphism. Static polymorphism is the feature by which an object is linked with the respective function or operator based on the values during the compile time. Static or Compile time Polymorphism can be achieved through Method overloading or operator overloading.

31. What is meant by dynamic polymorphism?

Dynamic Polymorphism or Runtime polymorphism refers to the type of Polymorphism in OOPs, by which the actual implementation of the function is decided during the runtime or execution. The dynamic or runtime polymorphism can be achieved with the help of method overriding.

32. What is the difference between overloading and overriding?

Overloading is a compile-time polymorphism feature in which an entity has multiple implementations with the same name. For example, Method overloading and Operator overloading.

Whereas Overriding is a runtime polymorphism feature in which an entity has the same name, but its implementation changes during execution. For example, Method overriding.
Image

33. How is data abstraction accomplished?

Data abstraction is accomplished with the help of abstract methods or abstract classes.

34. What is an abstract class?

An abstract class is a special class containing abstract methods. The significance of abstract class is that the abstract methods inside it are not implemented and only declared. So as a result, when a subclass inherits the abstract class and needs to use its abstract methods, they need to define and implement them.

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

Interface and abstract class both are special types of classes that contain only the methods declaration and not their implementation. But the interface is entirely different from an abstract class. The main difference between the two is that, when an interface is implemented, the subclass must define all its methods and provide its implementation. Whereas when an abstract class is inherited, the subclass does not need to provide the definition of its abstract method, until and unless the subclass is using it.

Also, an abstract class can contain abstract methods as well as non-abstract methods.

36. What are access specifiers and what is their significance?

Access specifiers, as the name suggests, are a special type of keywords, which are used to control or specify the accessibility of entities like classes, methods, etc. Some of the access specifiers or access modifiers include “private”, “public”, etc. These access specifiers also play a very vital role in achieving Encapsulation - one of the major features of OOPs.

37. What is an exception?

An exception can be considered as a special event, which is raised during the execution of a program at runtime, that brings the execution to a halt. The reason for the exception is mainly due to a position in the program, where the user wants to do something for which the program is not specified, like undesirable input.

38. What is meant by exception handling?

No one wants its software to fail or crash. Exceptions are the major reason for software failure. The exceptions can be handled in the program beforehand and prevent the execution from stopping. This is known as exception handling.
So exception handling is the mechanism for identifying the undesirable states that the program can reach and specifying the desirable outcomes of such states.
Try-catch is the most common method used for handling exceptions in the program.

39. What is meant by Garbage Collection in OOPs world?

Object-oriented programming revolves around entities like objects. Each object consumes memory and there can be multiple objects of a class. So if these objects and their memories are not handled properly, then it might lead to certain memory-related errors and the system might fail.

Garbage collection refers to this mechanism of handling the memory in the program. Through garbage collection, the unwanted memory is freed up by removing the objects that are no longer needed.

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

No. Java applications are based on Object-oriented programming models or OOPs concept, and hence they cannot be implemented without it.

However, on the other hand, C++ can be implemented without OOPs, as it also supports the C-like structural programming model.

OOPs Coding Problems

41. What is the output of the below code?

```
#include<iostream>

using namespace std;
class BaseClass1 {
public:
    BaseClass1()
    { cout << " BaseClass1 constructor called" << endl;  }
};

class BaseClass2 {
public:
    BaseClass2()
    { cout << "BaseClass2 constructor called" << endl;  }
};

class DerivedClass: public BaseClass1, public BaseClass2 {
  public:
   DerivedClass()
    {  cout << "DerivedClass constructor called" << endl;  }
};

int main()
{
  DerivedClass derived_class;
  return 0;
}
```

Output:

```
BaseClass1 constructor called
BaseClass2 constructor called
DerivedClass constructor called
```

Reason:
The above program demonstrates Multiple inheritances. So when the Derived class's constructor is called, it automatically calls the Base class's constructors from left to right order of inheritance.

42. What will be the output of the below code?

```
class Scaler
{
  static int i;

  static
  {
     System.out.println("a");

     i = 100;
  }
}

public class StaticBlock
{
  static
  {
     System.out.println("b");
  }

  public static void main(String[] args)
  {
     System.out.println("c");

     System.out.println(Scaler.i);
  }
}
```

Output:

```
b
c
a
100
```

Reason:
Firstly the static block inside the main-method calling class will be implemented. Hence 'b' will be printed first. Then the main method is called, and now the sequence is kept as expected.

43. Predict the output?

```
#include<iostream>
using namespace std;

class ClassA {
public:
   ClassA(int ii = 0) : i(ii) {}
   void show() { cout << "i = " << i << endl;}
private:
   int i;
};

class ClassB {
public:
   ClassB(int xx) : x(xx) {}
   operator ClassA() const { return ClassA(x); }
private:
   int x;
};

void g(ClassA a)
{  a.show(); }

int main() {
 ClassB b(10);
 g(b);
 g(20);
 getchar();
 return 0;
}
```

Output:

```
i = 10
i = 20
```

Reason:
ClassA contains a conversion constructor. Due to this, the objects of ClassA can have integer values. So the statement g(20) works. Also, ClassB has a conversion operator overloaded. So the statement g(b) also works.

44. What will be the output in below code?

```
public class Demo{
   public static void main(String[] arr){
         System.out.println(“Main1”);
   }
   public static void main(String arr){
         System.out.println(“Main2”);
   }
}
```

Output:

```
Main1
```

Reason:
Here the main() method is overloaded. But JVM only understands the main method which has a String[] argument in its definition. Hence Main1 is printed and the overloaded main method is ignored.

45. Predict the output?

```
#include<iostream>
using namespace std;

class BaseClass{
   int arr[10];
};

class DerivedBaseClass1: public BaseClass { };

class DerivedBaseClass2: public BaseClass { };

class DerivedClass: public DerivedBaseClass1, public DerivedBaseClass2{};

int main(void)
{
 cout<<sizeof(DerivedClass);
 return 0;
}
```

Output:

```
If the size of the integer is 4 bytes, then the output will be 80.
```

Reason:
Since DerivedBaseClass1 and DerivedBaseClass1 both inherit from class BaseClass, DerivedClass contains two copies of BaseClass. Hence it results in wastage of space and a large size output. It can be reduced with the help of a virtual base class.

46. What is the output of the below program?

```
#include<iostream>

using namespace std;
class A {
public:
  void print()
  { cout <<" Inside A::"; }
};

class B : public A {
public:
  void print()
  { cout <<" Inside B"; }
};

class C: public B {
};

int main(void)
{
 C c;

 c.print();
 return 0;
}
```

Output:

```
Inside B
```

Reason:
The above program implements a Multi-level hierarchy. So the program is linearly searched up until a matching function is found. Here, it is present in both classes A and B. So class B’s print() method is called.

1. What is the difference between OOP and SOP?

Object-Oriented Programming	Structural Programming
Object-Oriented Programming is a type of programming which is based on objects rather than just functions and procedures	Provides logical structure to a program where programs are divided functions
Bottom-up approach	Top-down approach
Provides data hiding	Does not provide data hiding
Can solve problems of any complexity	Can solve moderate problems
Code can be reused thereby reducing redundancy	Does not support code reusability

2. What is Object Oriented Programming?

Object-Oriented Programming(OOPs) is a type of programming that is based on objects rather than just functions and procedures. Individual objects are grouped into classes. OOPs implements real-world entities like inheritance, polymorphism, hiding, etc into programming. It also allows binding data and code together.

3. Why use OOPs?

- OOPs allows clarity in programming thereby allowing simplicity in solving complex problems
- Code can be reused through inheritance thereby reducing redundancy
- Data and code are bound together by encapsulation
- OOPs allows data hiding, therefore, private data is kept confidential
- Problems can be divided into different parts making it simple to solve
- The concept of polymorphism gives flexibility to the program by allowing the entities to have multiple forms

4. What are the main features of OOPs?

- Inheritance
- Encapsulation
- Polymorphism
- Data Abstraction

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

Class: User-defined blueprint from which objects are created. It consists of methods or set of instructions that are to be performed on the objects.

Structure: A structure is basically a user-defined collection of variables which are of different data types.

8. Can you call the base class method without creating an instance?

Yes, you can call the base class without instantiating it if:

- It is a static method
- The base class is inherited by some other subclass

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

Object	Class
A real-world entity which is an instance of a class	A class is basically a template or a blueprint within which objects can be created
An object acts like a variable of the class	Binds methods and data together into a single unit
An object is a physical entity	A class is a logical entity
Objects take memory space when they are created	A class does not take memory space when created
Objects can be declared as and when required	Classes are declared just once

10. What is inheritance?
Inheritance is a feature of OOPs which allows classes inherit common properties from other classes. For example, if there is a class such as 'vehicle', other classes like 'car', 'bike', etc can inherit common properties from the vehicle class. This property helps you get rid of redundant code thereby reducing the overall size of the code.

11. What are the different types of inheritance?

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

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

Multiple Inheritance	Multilevel Inheritance
Multiple inheritance comes into picture when a class inherits more than one base class	Multilevel inheritance means a class inherits from another class which itself is a subclass of some other base class
Example: A class defining a child inherits from two base classes Mother and Father	Example: A class describing a sports car will inherit from a base class Car which inturn inherits another class Vehicle

13. What is hybrid inheritance?
Hybrid inheritance is a combination of multiple and multi-level inheritance.

14. What is hierarchical inheritance?
Hierarchical inheritance refers to inheritance where one base class has more than one subclasses. For example, the vehicle class can have 'car', 'bike', etc as its subclasses.

15. What are the limitations of inheritance?

- Increases the time and effort required to execute a program as it requires jumping back and forth between different classes
- The parent class and the child class get tightly coupled
- Any modifications to the program would require changes both in the parent as well as the child class
- Needs careful implementation else would lead to incorrect results

16. What is a superclass?
A superclass or base class is a class that acts as a parent to some other class or classes. For example, the Vehicle class is a superclass of class Car.

17. What is a subclass?
A class that inherits from another class is called the subclass. For example, the class Car is a subclass or a derived of Vehicle class.

18. What is polymorphism?

Polymorphism refers to the ability to exist in multiple forms. Multiple definitions can be given to a single interface. For example, if you have a class named Vehicle, it can have a method named speed but you cannot define it because different vehicles have different speed. This method will be defined in the subclasses with different definitions for different vehicles.

19. What is static polymorphism?

Static polymorphism (static binding) is a kind of polymorphism that occurs at compile time. An example of compile-time polymorphism is method overloading.

20. What is dynamic polymorphism?

Runtime polymorphism or dynamic polymorphism (dynamic binding) is a type of polymorphism which is resolved during runtime. An example of runtime polymorphism is method overriding.

21. What is method overloading?

Method overloading is a feature of OOPs which makes it possible to give the same name to more than one methods within a class if the arguments passed differ.

22. What is method overriding?

Method overriding is a feature of OOPs by which the child class or the subclass can redefine methods present in the base class or parent class. Here, the method that is overridden has the same name as well as the signature meaning the arguments passed and the return type.

23. What is operator overloading?

Operator overloading refers to implementing operators using user-defined types based on the arguments passed along with it.

24. Differentiate between overloading and overriding.

Overloading	Overriding
Two or more methods having the same name but different parameters or signature	Child class redefining methods present in the base class with the same parameters/ signature
Resolved during compile-time	Resolved during runtime

25. What is encapsulation?

Encapsulation refers to binding the data and the code that works on that together in a single unit. For example, a class. Encapsulation also allows data-hiding as the data specified in one class is hidden from other classes.

26. What are ‘access specifiers’?

Access specifiers or access modifiers are keywords that determine the accessibility of methods, classes, etc in OOPs. These access specifiers allow the implementation of encapsulation. The most common access specifiers are public, private and protected. However, there are a few more which are specific to the programming languages.

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

Name	Accessibility from own class	Accessibility from derived class	Accessibility from world
Public	Yes	Yes	Yes
Private	Yes	No	No
Protected	Yes	Yes	No

28. What is data abstraction?

Data abstraction is a very important feature of OOPs that allows displaying only the important information and hiding the implementation details. For example, while riding a bike, you know that if you raise the accelerator, the speed will increase, but you don’t know how it actually happens. This is data abstraction as the implementation details are hidden from the rider.

29. How to achieve data abstraction?

Data abstraction can be achieved through:

- Abstract class
- Abstract method

30. What is an abstract class?

An abstract class is a class that consists of abstract methods. These methods are basically declared but not defined. If these methods are to be used in some subclass, they need to be exclusively defined in the subclass.

31. Can you create an instance of an abstract class?

No. Instances of an abstract class cannot be created because it does not have a complete implementation. However, instances of subclass inheriting the abstract class can be created.

32. What is an interface?

It is a concept of OOPs that allows you to declare methods without defining them. Interfaces, unlike classes, are not blueprints because they do not contain detailed instructions or actions to be performed. Any class that implements an interface defines the methods of the interface.

33. Differentiate between data abstraction and encapsulation.

Data abstraction	Encapsulation
Solves the problem at the design level	Solves the problem at the implementation level
Allows showing important aspects while hiding implementation details	Binds code and data together into a single unit and hides it from the world

34. What are virtual functions?

Virtual functions are functions that are present in the parent class and are overridden by the subclass. These functions are used to achieve runtime polymorphism.

35. What are pure virtual functions?

Pure virtual functions or abstract functions are functions that are only declared in the base class. This means that they do not contain any definition in the base class and need to be redefined in the subclass.

36. What is a constructor?

A constructor is a special type of method that has the same name as the class and is used to initialize objects of that class.

37. What is a destructor?

A destructor is a method that is automatically invoked when an object is destroyed. The destructor also recovers the heap space that was allocated to the destroyed object, closes the files and database connections of the object, etc.

38. Types of constructors

Types of constructors differ from language to language. However, all the possible constructors are:

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

39. What is a copy constructor?

A copy constructor creates objects by copying variables from another object of the same class. The main aim of a copy constructor is to create a new object from an existing one.

40. What is the use of 'finalize'?

Finalize as an object method used to free up unmanaged resources and cleanup before Garbage Collection(GC). It performs memory management tasks.

41. What is Garbage Collection(GC)?

GC is an implementation of automatic memory management. The Garbage collector frees up space occupied by objects that are no longer in existence.

42. Differentiate between a class and a method.

Class	Method
A class is basically a template that binds the code and data together into a single unit. Classes consist of methods, variables, etc	Callable set of instructions also called a procedure or function that are to be performed on the given data

43. Differentiate between an abstract class and an interface?

Basis for comparison	Abstract Class	Interface
Methods	Can have abstract as well as other methods	Only abstract methods
Final Variables	May contain final and non-final variables	Variables declared are final by default
Accessibility of Data Members	Can be private, public, etc	Public by default
Implementation	Can provide the implementation of an interface	Cannot provide the implementation of an abstract class

44. What is a final variable?

A variable whose value does not change. It always refers to the same object by the property of non-transversity.

45. What is an exception?

An exception is a kind of notification that interrupts the normal execution of a program. Exceptions provide a pattern to the error and transfer the error to the exception handler to resolve it. The state of the program is saved as soon as an exception is raised.

46. What is exception handling?

Exception handling in Object-Oriented Programming is a very important concept that is used to manage errors. An exception handler allows errors to be thrown and caught and implements a centralized mechanism to resolve them.

47. What is the difference between an error and an exception?

Error	Exception
Errors are problems that should not be encountered by applications	Conditions that an application might try to catch

48. What is a try/ catch block?

A try/ catch block is used to handle exceptions. The try block defines a set of statements that may lead to an error. The catch block basically catches the exception.

49. What is a finally block?

A finally block consists of code that is used to execute important code such as closing a connection, etc. This block executes when the try block exits. It also makes sure that finally block executes even in case some unexpected exception is encountered.

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

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 to 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) Explain the term constructor

A 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 the same as a class name.
- A constructor must have no return type.

10) Define Destructor?

A 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 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.

12) What is a virtual function?

A virtual function is a member function of a 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.

A virtual function can be declared using a token(virtual) in C++. It can be achieved in C/Python Language by using function pointers or pointers to function.

13) 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.

A 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 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.

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 its own precedence to execute

16) What is an abstract class?

An abstract class is a class which cannot be instantiated. Creation of an object is not possible with an abstract class, but it can be inherited. An abstract class can contain only an Abstract method. Java allows only abstract method in abstract class while other languages allow non-abstract method as well.

17) What is a ternary operator?

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

20) What is the super keyword?

The super keyword is used to invoke the 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 a subclass to provide the implementation of a method that overrides in the main class. It will override 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 an abstract method. If the class implements an interface, it thereby inherits all the abstract methods of an interface.

Java uses Interface to implement multiple inheritances.

23) What is exception handling?

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

24) What are tokens?

A compiler recognizes a token, 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. Example: Brackets, Commas, Braces, and Parentheses.

25) What is the main 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 equal value in the same class itself.

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

26) What is the main difference between a class and an object?

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

A class can have sub-classes, while an object doesn't have sub-objects.

27) What is an abstraction?

Abstraction is a useful feature of OOPS, and it shows only the necessary details to the client of an object. Meaning, it shows only required details for an object, not the inner constructors, of an object. Example – When you want to switch on the television, it is not necessary to know the inner circuitry/mechanism needed to switch on the TV. Whatever is required to switch on TV will be shown by using an abstract class.

28) What are the access modifiers?

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

- Private
- Protected
- Public
- Friend
- Protected Friend

29) What are sealed modifiers?

Sealed modifiers are the access modifiers where the methods can not inherit it. Sealed modifiers can also be applied to properties, events, and methods. This modifier cannot be used 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 a 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 types of constructors:

– 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 the assignment of values to variables during design time, whereas late Binding refers to the assignment of values to variables during run time.

35) What is the difference between structure and a class?

The default access type of a Structure is public, but class access type is private. A structure is used for grouping data, whereas a 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 encapsulate and inherent data, which requires strict validation.

36) What is the default access modifier in a class?

The default access modifier of a class is Internal and the default access modifier of a class member is Private.

37) What is a 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 a parameter for constructors?

No, we do not require a 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 always be only one 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 nonstatic members?

False.

44) What are a base class, subclass, and superclass?

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

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

The superclass 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. In other words, you cannot create an instance of 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 a reuse mechanism?

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

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

Encapsulation

1) Define structured programming and its disadvantage?

It is a program design technique. In structured programming languages like C, Pascal, programmer defines data structure (arrays, structures, unions, enum etc) and the functions that perform operations on defined data structures.

But when program size grows up, then it becomes un-manageable, and data accidently modified by the different functions thus it generates logical errors or bugs in program.

3) Define class?

Class is a logical encapsulation of data members and member functions in a single unit. It is a template of object. Class does not occupy any space in memory but when object creates, it occupies space in the memory according to data member. An empty class takes 1 byte space in memory.

For example, a **HUMAN** is a class and person **"RAM"** and **"SHYAM"** are the objects.

4) What is an Encapsulation?

It is a one of the basic feature of OOPS. **Encapsulation means Binding data members and member functions in a single unit**. Encapsulation can be useful to keep data safe from outside interfaces.

5) What is an Inheritance?

Inheritance is a mechanism (also an important component/feature of object oriented programming system) to inherit features from one to another class.

If we want to use/access existing features of any class, we can access them by using Inheritance. There will two classes **Base class** and **Derived class**.

If there is an existing class named **"class_one"** and new class
named **"class_two"** that will access the features of class_one. In this
case **"class_one"** will be considered as Base class and **"class_two"** as Derived class.

6) What is a polymorphism?

Polymorphism is the most important concept of OOPS. Polymorphism provides ability to use same name for different purposes.

In C++, functions and operators can be used to perform several (different) tasks by having same names. Two types of polymorphism are used:

1. Static or Compile time polymorphism
   0. Function Overloading
   1. Operator Overloading
2. Dynamic or Runtime polymorphism
   0. Virtual function or dynamic binding

7) What is an Abstraction?

Abstraction means hiding the implementation detail for simplicity. It is a good programming practice to keep implementation and interface independent. Abstraction allows doing the same.

We do not need to change interface, if we are going to change the implementation. Two types of abstractions are:

1. Function abstraction
2. Data abstraction

8) What is the difference between object based language and object oriented language?

OOBS does not support inheritance and polymorphism, whereas OOPS supports class, object, polymorphism, inheritance, abstraction etc.

9) What is the basic difference between Visual Basic and Visual C++?

Visual Basic is an object-based language; whereas Visual C++ is an object-oriented language.

10) What is the advantage of C++ being a block-structured language?

As we know that memory space is always premium. In block structured language, we can save memory space by controlling the life of variables.

For example:

```
void fun()

{

   //50 statement

   {

      int x;



      //statement using x

   }

   //100 more statement.
```

In the inner block, there is a variable x which will be declared when program's control reaches to that block and freed when program's control goes from that block. Thus it will not keep the memory save for a longtime.

1. What is OOPs?

OOPs stands for representing the Object-Oriented Programming system. Programs are treated as a collection of objects in oops. Each object is nothing but an example of a class.

2. Difference between Procedural programming and OOPs?

Procedural Programming:

Procedural Programming: is based on functions.

-It shows the data to the entire program.
-It does not have scope for code reuse.
-It follows a the concept of top-down programming.
-Nature of the language is complicated.
-It is hard to modify, extend and maintain the code.

Object-oriented programming:
-It is based on real-world objects.
-It encapsulates the data.
-It provides more scope of code reuse.
-It follows a bottom-up programming paradigm.
-It is less complicated in nature, so it is easier to modify, extend and maintain.

3. Why use OOPs?

OOPs has clarity in programming. It has flexibility and simplicity in solving complex problems. Reuse of code is easy as the Inheritance concept helps to reduce the redundancy of code. Data and code are bound together by encapsulation. OOPs has features for data hiding, so private data can be stored and maintain confidentiality. Problems can be divided into different parts making it simple to solve. The concept of polymorphism has flexibility for a single entity can have multiple forms.

5. What is Encapsulation?

Encapsulation is also a part of OOPs concept. It refers to the bundling of data with the methods that operate on that data. It also helps to restrict any direct access to some of an object’s components.

6. What is Abstraction?

Abstraction is an OOPs concept to build the structure of the real-world objects. It “shows” only essential attributes and “hides” unnecessary information from the outside. The main focus of **abstraction** is to hide the unnecessary details from the users. It is one of the most important concepts of **OOPs**.

7. What is method overloading?

There is a concept where two or more methods can have the same name. But they should have different parameters, different numbers of parameters, different types of parameters, or both. These methods are known as overloaded methods and this feature is called **method overloading**.

8. What is method overriding?

Method overriding is a concept of object-oriented programming.

It is a language feature that allows a subclass or child class to provide a specific implementation of a **method** which is already provided by one of its super classes or parent classes.

9. Types of Inheritance in OOPS

Hybrid Inheritance
Multiple Inheritance
Single Inheritance
Multi-level Inheritance
Hierarchical Inheritance

10. What is an object?

Object: An object is an instance of a class and also It has its own identity and behaviour.

11. What is Method?

It basically goes for describing the set of instructions and it is also called a procedure.

12. What is a class?

Class is a kind of a user-defined data type that contains variables, properties, and methods. It also helps to find the properties of an object.

13. What are constructors?

The constructor has the same name as the class.
A constructor is also a special kind of method. It is used to initialize objects of the class.

14. Types of constructor

Types of constructor depend upon languages

Private Constructor
Default Constructor
Copy Constructor
Static Constructor
Parameterized Constructor

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

Class: Class is basically a User-defined blueprint from which objects are created. It consists of methods (set of instructions) that are performed on the objects.

Structure: A structure is also a user-defined collection of variables. Structures are also different data types.

16. What are the access modifiers?

Access modifiers or access specifiers are the keywords in object-oriented languages. It helps to set the accessibility of **classes**, **methods**, and other members.

17. What are languages come under the **oops concept?**

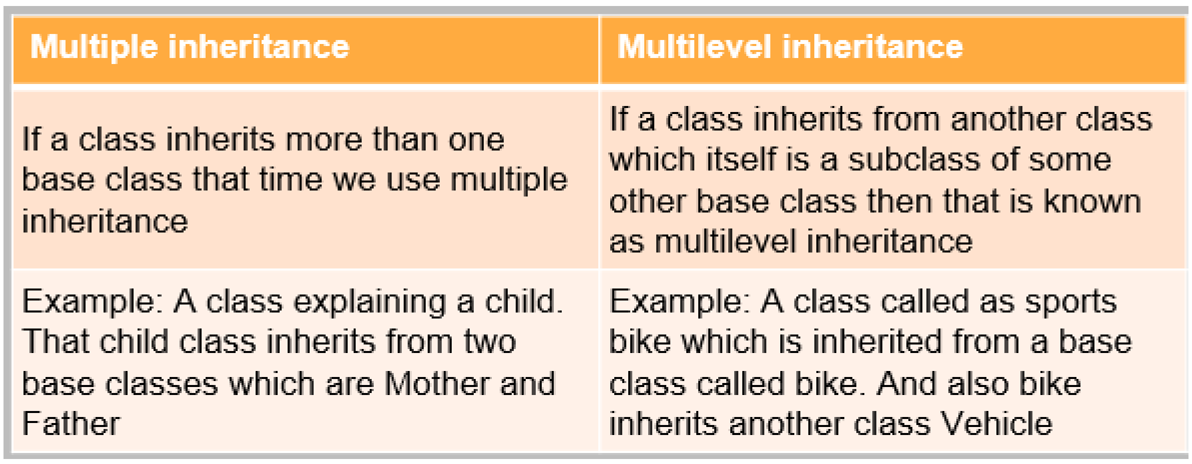
Simula is known as the first object-oriented programming language, the most popular OOP languages are:

Java
JavaScript
Python
C++
Visual Basic . NET.
Ruby
Scala
PHP

18. What is inheritance?

If you derive a class from another class that is known as inheritance.

The child class will **inherit** all the public and protected properties and methods from the parent class. The child class can also have its own properties and methods. An **inherited** class is defined by using the extends keyword.



Multiple inheritance	Multilevel inheritance
If a class inherits more than one base class that time we use multiple inheritance	If a class inherits from another class which itself is a subclass of some other base class then that is known as multilevel inheritance
Example: A class explaining a child. That child class inherits from two base classes which are Mother and Father	Example: A class called as sports bike which is inherited from a base class called bike. And also bike inherits another class Vehicle

19. What is hybrid inheritance?

A combination of multiple and multi-level inheritances is known as hybrid inheritance.

20. What is hierarchical inheritance?

Defining Hierarchical inheritance is basically when one base class has more than one subclasses. For example, the fruit class can have ‘apple’, ’mango’, ’banana’, ‘cherry’ etc. as its subclasses.

21. What are the limitations of inheritance?

It Increases the execution time and effort. It also requires jumping back and forth between different classes. The parent class and the child class are always tightly coupled. Afford modifications in the program would require changes for the parent and the child’s class. Inheritance requires careful implementation otherwise it would lead to incorrect results.

22. What is a superclass?

A superclass or base class is also a class that works as a parent to some other class/ classes.

For example, the Vehicle class is a superclass of class Bike.

23. What is a subclass?

A subclass is a class that inherits from another class. For example, the class Bike is a subclass or a derive of the Vehicle class.

24. What is Polymorphism?

Polymorphism is one of the most used and the core concepts in **OOP** languages. It explains the concept of different classes can be used with the same interface. Each of these classes can have its own implementation of the interface.

25. What is static polymorphism?

Static polymorphism or static binding is one kind of polymorphism that comes at compile time. An example of compile-time polymorphism is: method overloading.

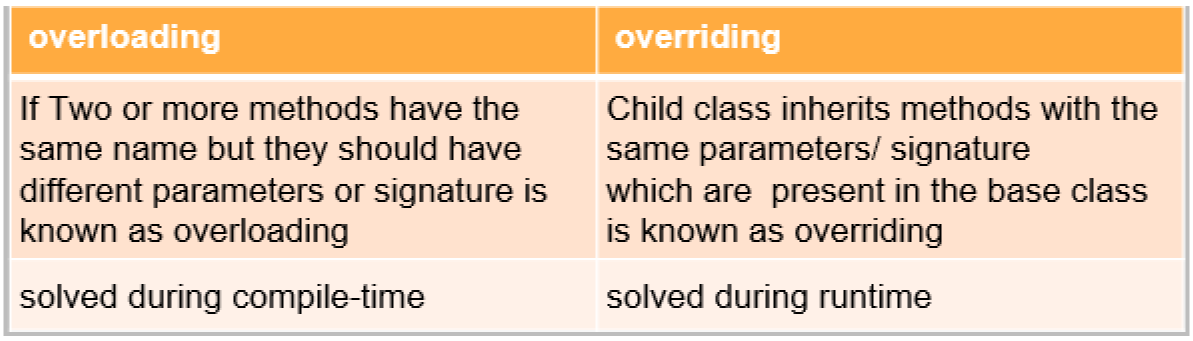
26. What is dynamic polymorphism?

Dynamic polymorphism, dynamic binding or Runtime polymorphism is also part of polymorphism which is basically solved during runtime. An example of runtime polymorphism: method overriding.

27. What is operator overloading?

Operator overloading is used to implement operators using user-defined types, based on the arguments passed along with it.

28. Differentiate between overloading and overriding.

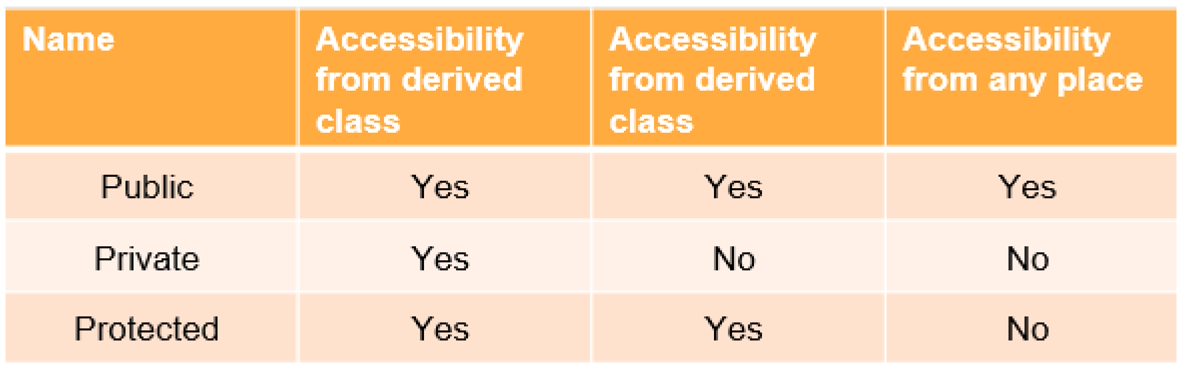


overloading	overriding
If Two or more methods have the same name but they should have different parameters or signature is known as overloading	Child class inherits methods with the same parameters/ signature which are present in the base class is known as overriding
solved during compile-time	solved during runtime

29. What is encapsulation?

Encapsulation is used to wrap the data and the code which works on in a single unit together. Example: Encapsulation allows data-hiding as the data specified in one class is hidden from other classes.

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



Name	Accessibility from derived class	Accessibility from derived class	Accessibility from any place
Public	Yes	Yes	Yes
Private	Yes	No	No
Protected	Yes	Yes	No

31. What is data abstraction?

Data abstraction is one of the most important features of OOPs. It only allows important information to be displayed. It helps to hide the implementation details.

For example, while using a mobile, you know, how can you message or call someone but you don't know how it actually happens.

This is data abstraction as the implementation details are hidden from the user.

32. How to achieve data abstraction?

Data abstraction can be achieved using two ways:

Abstract class
Abstract method

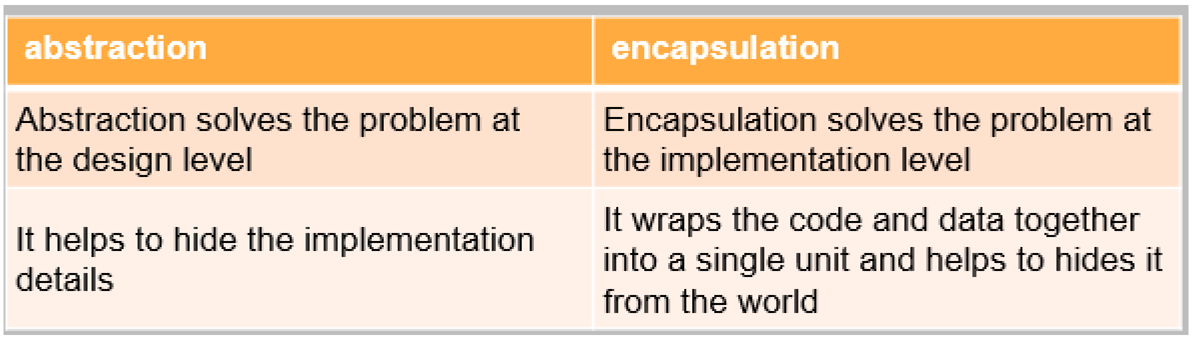
33. What is an abstract class?

An abstract class is also a class which is consists of abstract methods.

So what is abstract method?

These methods are basically declared but not defined and If these methods need to be used later in some subclass that time those methods haveto be exclusively defined in the subclass.

34. Differentiate between data abstraction and encapsulation.



abstraction	encapsulation
Abstraction solves the problem at the design level	Encapsulation solves the problem at the implementation level
It helps to hide the implementation details	It wraps the code and data together into a single unit and helps to hides it from the world

35. What are virtual functions?

Virtual functions are also part of the functions which are present in the parent class and they are overridden by the subclass.

These functions help to achieve runtime polymorphism.

36. What is a destructor?

A destructor is a method which is called automatically when an object is destroyed.

The destructor also recovers the heap space which was allocated to the destroyed object. It also start closing the files and database connections of the object, etc.

37. What is a copy constructor?

A copy constructor basically creates objects by copying variables from another object from the same class. The main focus of a copy constructor is to make a new object from an existing

38. What is the use of ‘finalize’?

Finalize is used to free the unmanaged resources and also help to clean before Garbage Collection(GC). It performs memory management tasks.

39. What is Garbage Collection(GC)?

Garbage Collection is a part of automatic memory management. The Garbage collector helps to free the occupied spaces by objects. Those spaces are no longer in existence.

40. What is a final variable?

The final variable does not change and It always refers to the same object by the property of non-transversity.

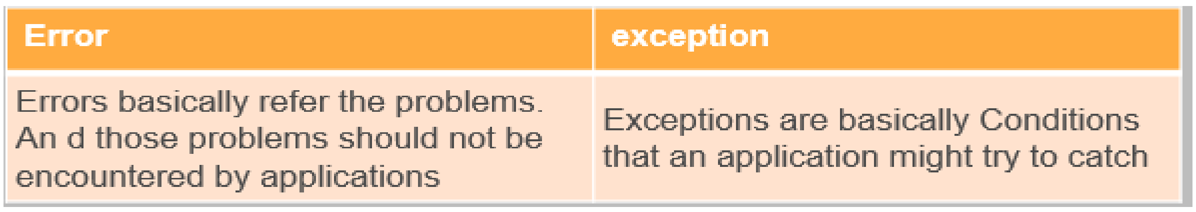
41. What is an exception?

An exception is a kind of message that interrupts and comes up when there is an issue with normal execution of a program. Exceptions provide a error and transfer that error to the exception handler to resolve it. The state of the program is saved as soon as an exception is raised.

42. What is exception handling?

Exception handling in Object-Oriented Programming is the most important concept. It is used to manage errors. An exception handler help to throw errors and then catch the error in order to solve them.

43. What is the difference between an error and an exception?



Error	exception
Errors basically refer the problems. An d those problems should not be encountered by applications	Exceptions are basically Conditions that an application might try to catch

44. What is a try/ catch block?

A try/ catch block helps to handle exceptions. The try block explains a set of statements that may have an error. The catch block basically catches the exception.

45. What is a finally block?

A finally block executes when the try block exits and It also executes even in case some unexpected exception is encountered. Finally block normally consists of some important part of the program.

46. Can you call the base class method without creating an instance?

•Yes, you are allowed to call the base class without instantiating it but there are some conditions that are applicable:

•If it is a static method

•The base class is inherited by some other subclass

47. What is the difference between OOP and SOP?

Object-oriented programming involves concepts of objects and classes. Everything is considered as an object which has specific properties and behaviors which are represented in a class. Object-oriented programming provides encapsulation and abstraction in the code. Ex: – Java Programming language.

Structure-oriented programming involves the concepts of functions and structures. Everything is considered functionality and structures, represented using functions— Ex: – C Programming language.

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

Any real-world entity is called an object. The object has specific properties and behaviors, and the similar type of objects having similar features and behaviors are grouped as a class. Hence, Class is a blueprint of objects, and an object is an instance of a class.

Ex: – 1. An Animal is a class, and cat, dog, etc., are objects with common properties like name, type, and common behaviors like speaking, walking, running, etc.

2. Mobile is a class, and Nokia, moto, iPhone, etc., are objects with common properties like modal_no, color, etc., and common behaviors like audio_calling, video_calling, music, etc.

49. What are 'access specifiers'?

Access specifiers are the keywords in any programming language used to specify the Class's, method's, interface's and variable's behavior concerning its accessibility. The access specifiers in C++ Programming are public, private, and protected.

50. Can you create an instance of an abstract class?

No, an instance of Abstract class cannot be created. To implement the abstract Class, abstract methods, the Abstract Class should be extended by another class, and the object of the implementation class can be created.

51. What is an interface?

An interface is a user-defined data type and is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface. A class describes an object's attributes and behaviors, and an interface contains behaviors that a class implements. The Class represents “how,” and the interface represents “what’.

52. What are pure virtual functions?

A pure virtual function/method is a function whose implementations are not provided in the base class, and only a declaration is provided. The pure virtual function can have its implementation code in the derived class; otherwise, the derived class will also be considered an abstract Class. The Class containing pure virtual functions is abstract.

53. Differentiate between a class and a method.

A class is a blueprint of objects, and it consists of the properties and behavior of the objects.

Methods are programming constructs that perform specific tasks/behavior.

54. Differentiate between an abstract class and an interface?

An interface can have only abstract methods, but an Abstract class can have abstract and non-abstract methods.

The interface should be used if just the requirement specification is known and nothing about implementation. If the implementation is known, but partially, then abstract Class should be used. If the implementation is known completely, then concrete Class should be used.

55. What are the limitations of OOPs?

1. Larger Program size – Programs can become lengthy if written using object-oriented programming concepts compared to procedure-oriented programming.

2. Slower execution – As the number of lines of code to be executed is more comparatively, the execution time is also more.

3. Not suitable for all types of Problems.

4. Testing time is also higher for OOP Solutions.

56.What are the characteristics of an abstract class?

1. A class having at least one pure virtual function is called an Abstract class.

2. An Abstract class cannot have objects created, i.e., an abstract class cannot be instantiated, but Object references can be created.

3. An Abstract class can have non-abstract functions and pure virtual functions also.

4. The pure virtual function can have its implementation code in the derived class; otherwise, the derived class will also be considered an abstract Class

57. What is constructor chaining?

Constructor chaining is a method to call one constructor from another concerning current object reference. It can be done in two ways: –

1. Using the "this" keyword, the reference can be made to the constructor in the current class.

2. To call the constructor from the base class "super" keyword will be used.

58.What is Coupling in OOP, and why is it helpful?

The degree of dependency between the components is called coupling.

Types of Coupling

a. Tight Coupling – If the dependency between components is high, these components are called tightly coupled.

Ex: –

Below three Classes are highly dependent on each other hence they are tightly coupled.

```
class P

{
```

```
static int a = Q.j;

}



class Q

{

static int j = R.method();

}



class R

{

public static int method(){

return 3;

}
```

b. Loose Coupling – If the dependency between components is low, it is called loose coupling. Loose coupling is preferred because of the following reasons:-

1. It increases the maintainability of code

2. It provides reusability of code

59.Name the operators that cannot be overloaded

All the operators except the + operator cannot be overloaded.

60. What is Cohesion in OOP?

The modules having well-defined and specific functionality is called cohesion.

Advantages

It improves the maintainability and reusability of code.

61.What are the levels of data abstraction?

Highlighting the set of services by hiding internal implementation details is called abstraction.

By using abstract Class and interface, we can implement abstraction

62.What are the types of variables in OOP?

Variables are basic units to store data in RAM for Java programs.

Variables should be declared before using it in Java programming. Variable initialization can be static or dynamic. The syntax for variable declaration and static initialization is: –

Types of variables

1. Primitive Variables

It is used to represent primitive values like int, float, etc.

2. Reference Variables

It is used to refer to objects in Java.

3. Instance Variables

Variables whose value varied from object to object are instance variables. For every object, a separate copy of the instance variable is created. Instance variables are declared within the Class and outside any method/block/constructor

4. Static variables

For static Variables, a single copy of the variable is created, and that copy is shared between every Class object. The static variable is created during class loading and destroyed at class unloading.

Static variables can be accessed directly from the static and instance area. We are not required to perform initialization explicitly for static variables, and JVM will provide default values.

5. Local Variables

Variables declared inside a method or block or constructor are local variables. Hence the scope of local variables is the same as the block's scope in which we declared that variable.

JVM doesn't provide default values, and before using that variable, the initialization should be performed explicitly.

1. Why do we need to use OOPs?

OOPs needs to be used for:

1. making programming clearer and problem-solving more concise
2. reusing code with the help of inheritance
3. reducing redundancy
4. encapsulation
5. data hiding
6. the division into subproblems
7. program flexibility using polymorphism

2. What is multiple inheritance?

If one class shares the behavior and structure defined in another multiple class, it is called multiple inheritance.

3. Give an example of encapsulation.

The notion of data hiding is referred to as encapsulation. Protected and private members in C++ are examples.

4. What is the difference between overloading and overriding?

Overloading is two or more methods having the same name but different parameters. It is solved during compile-time. Whereas, Overriding is an OOPs concept that allows sub-classes to have a specific implementation of a method already provided by its parent class. It is solved during runtime.

5. Define protected access modifier.

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

6. What is the function of a super keyword?

The super keyword 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.

7. 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.

8. 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.

9. 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.

10. 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.

1. 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.

2. 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.

3. 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.

4. Name three operators that can’t be overloaded.

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

5. 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.

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

Yes. The limitations of inheritance are:

1. Increased execution effort and time
2. Tight coupling of parent and child class
3. Requires correct implementation
4. Requires jumping between different classes

7. 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.

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

1. It requires intensive testing
2. Not apt for minor problems
3. It requires good planning
4. It takes more time to solve problems
5. Problems need to be thought in term of objects

9. 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.

10. 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