OOPS Interview Cheatsheet

Most Important Interview Topics for OOPS

1. Classes and Objects
2. Feature/characteristics of OOPs
3. Inheritance (Type and Mode)
4. Access Specifiers
5. Virtual (Functions and Class)
6. Abstract class and Interface
7. Compile-time and Runtime Polymorphism
8. Friend function and Friend class
9. Call by Value and Call by Reference
10. Abstraction
11. Exception Handling
12. Constructor and Destructor
13. Copy constructor
14. Copy assignment operator
15. Overloading (Function, Constructor, Operator)
16. Function Overriding
17. Reference variable
18. this pointer
19. const keyword
20. static keyword
21. virtual keyword
22. abstract keyword
23. final keyword
24. new keyword
25. super keyword
26. explicit keyword

Most Important Interview Questions for OOPS

1. What is Object-Oriented Programming?

OOPs refers to Object-Oriented Programming. It is the programming paradigm that is defined using objects. Objects can be considered as real-world instances of entities like class, that have some characteristics and behaviours.

2. Why OOP?

There are many reasons why OOPs is mostly preferred, but the most important among them are:

- OOPs helps users to understand the software easily, although they don't know the actual implementation.
- With OOPs, the readability, understandability, and maintainability of the code increase multifold.
- Even very big codebases can be easily written and managed easily using OOPs.

3. What are some major Object Oriented Programming languages?

The programming languages that use and follow the Object-Oriented Programming paradigm or OOPs, are known as Object-Oriented Programming languages. Some of the major Object-Oriented Programming languages include:

- Java
- C++
- Javascript
- Python
- PHP

And many more.

4. 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.

5. What are the main features of OOPs? [American Express] [Microsoft] [Intuit] [Oracle]

OOPs or Object Oriented Programming mainly comprises of the below four features:

- Inheritance
- Encapsulation
- Polymorphism
- Data Abstraction

6. What are some advantages of using OOPs?

- 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.

7. Why is OOPs so popular? [Oracle]

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. The main pillar of OOPs - Data Abstraction, Encapsulation, Inheritance, and Polymorphism, makes it easy for programmers to solve complex scenarios.

8. What is a class?

A class is a template or a blueprint, which contains some values, known as member data or member, and some set of rules, known as behaviours or functions. So when an object is created, it automatically takes the data and functions that are defined in the class.

9. What is an object?

An object refers to the instance of the class, which contains the instance of the members and behaviours 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 behaviour.

10. What is encapsulation?

Encapsulation is 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.

11. What is Polymorphism? [Microsoft] [Intuit]

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.

12. What are the two types of Polymorphism? [American Express]

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.

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.

13. 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.

14. What is meant by Inheritance? [Goldman Sachs]

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.

15. What is Abstraction?

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.

16. What are the levels of data abstraction?

To make the system efficient for retrieval of data and reduce the complexity of the users, developers use the method of Data Abstraction.
There are mainly three levels of data abstraction:

1. Internal Level: Actual PHYSICAL storage structure and access paths.
2. Conceptual or Logical Level: Structure and constraints for the entire database
3. External or View level: Describes various user views

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 initialise 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? [Adobe]

Constructors are special methods whose name is the same as the class name. The constructors serve the special purpose of initialising 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();`

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;
   }
}
```

- Parameterised constructor: The constructors that take some arguments are known as parameterised constructors.

```
 class ABC
{
   int x;

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

- Copy constructor: A copy constructor is a member function that initialises 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? [Oracle]

Contrary to constructors, which initialise 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. [American Express] [ServiceNow]

Inheritance is one of the major features of object-oriented programming, by which an entity inherits some characteristics and behaviours 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, 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? [American Express] [Microsoft]

The various types of inheritance include:

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

27. What is an Interface? [Goldman Sachs]

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.

28. 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.

29. 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.

30. What is function overloading? Explain with an example. [American Express]

Function overloading is a feature of object oriented programming where two or more functions can have the same name but different parameters.

```
#include <bits/stdc++.h>
using namespace std;

void print(int i) {
  cout << " Here is int " << i << endl;
}

void print(double  f) {
  cout << " Here is float " << f << endl;
}

void print(string s) {
  cout << " Here is string " << s << endl;
}

int main() {
  print(10);
  print(10.10);
  print("ten");
  return 0;
}
```

31. What is the difference between overloading and overriding? [American Express] [Adobe] [American Express] [Goldman Sachs] [Oracle]

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.

32. 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.

33. 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.

34. What is a friend function? [Adobe]

A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.
A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends.

```
class Box {
   double width;

   public:
      friend void printWidth( Box box );
      void setWidth( double wid );
};

// Member function definition
void Box::setWidth( double wid ) {
   width = wid;
}


void printWidth( Box box ) {
   /* Because printWidth() is a friend of Box, it can
   directly access any member of this class */
   cout << "Width of box : " << box.width <<endl;
}
```

```
// Main function for the program
int main() {
   Box box;

   // set box width without member function
   box.setWidth(10.0);

   // Use friend function to print the wdith.
   printWidth( box );

   return 0;
}
```

35. 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.

36. 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.

37. What is meant by exception handling?

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.

38. 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.

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

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

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

Yes, C++ can be implemented without OOPs, as it supports the C-like structural programming model.

41. What is a virtual function? [American Express] [Goldman Sachs] [Adobe] [Microsoft]

A virtual function is a member function which is declared within a base class and is re-defined(overriden) by a derived class. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the function.

42. What is a pure virtual function? [Adobe] [Microsoft]

A pure virtual function (or abstract function) is a virtual function for which we can have implementation, but we must override that function in the derived class, otherwise the derived class will also become abstract class.

43. What is a final variable?

If you make any variable as final, you cannot change the value of final variable, i.e., it will be constant.

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

Garbage means unreferenced objects and Garbage Collection is a process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects. To do so, we use free() function in C language, delete() in C++, but in Java, it is performed automatically.

45. What is composition?

Composition is a way to design or implement the **"has-a"** relationship. The composition relationship of two objects is possible when one object contains another object, and that object is fully dependent on it. The contained object should not exist without the existence of its parent object.
In a simple way, we can say it is a technique through which we can describe the reference between two or more classes. And for that, we use the instance variable, which should be created before it is used.

46. What is copy assignment operator?

The *copy assignment operator* lets you create a new object from an existing one by initialisation. A copy assignment operator of a class `A` is a non-static non-template member function that has one of the following forms:

- `A::operator=(A)`
- `A::operator=(A&)`
- `A::operator=(const A&)`
- `A::operator=(volatile A&)`
- `A::operator=(const volatile A&)`

47. Name the operators that cannot be overloaded.

In C++, following operators can not be overloaded:

1. . (Member Access or Dot operator)
2. ?: (Ternary or Conditional Operator)
3. :: (Scope Resolution Operator)
4. .* (Pointer-to-member Operator)
5. sizeof (Object size Operator)
6. typeid (Object type Operator)

48. What are Manipulators?

Manipulators are helping functions that can modify the input/output stream. It does not mean that we change the value of a variable, it only modifies the I/O stream using insertion (<<) and extraction (>>) operators. For example, if we want to print the hexadecimal value of 100 then we can print it as:

```
cout<<setbase(16)<<100;
```

49. What is const keyword?

The const keyword specifies that a variable's value is constant and tells the compiler to prevent the programmer from modifying it.

50. Why is virtual keyword used?

The virtual keyword is used to modify a method, property, indexer, or event declaration and allow for it to be overridden in a derived class.

51. Why is explicit keyword used?

If a class has a constructor which can be called with a single argument, then this constructor becomes conversion constructor because such a constructor allows conversion of the single argument to the class being constructed.
We can avoid such implicit conversions as these may lead to unexpected results. We can make the constructor explicit with the help of explicit keyword.

52. Why is static keyword used?

Static is a non-access modifier. It means that something (a field, method, block or nested class) is related to the type rather than any particular instance of the type.
The Static keyword can be applied to:
Static Method
Static Variable
Initialisation Block
Nested class

53. Why is super keyword used?

The **super** keyword in Java is a reference variable which is used to refer immediate parent class object.
Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

54. Why is this keyword used?

this is a **reference variable** that refers to the current object.

55. Why is new keyword used?

new keyword is used to create an instance of the class. In other words, it instantiates a class by allocating memory for a new object and returning a reference to that memory.

```
Class student=new Class();
```

Most Important Object Oriented Design Interview Questions

1. Design a Hashmap. [Goldman Sachs] [Walmart]
2. Design an ATM Machine. [American Express]
3. Design a Library Management System.
4. Design an online Coding Platform like LeetCode. [Flipkart]
5. Design a Car Rental System. [Flipkart]
6. Design a 2-Player Chess Game. [Amazon]
7. Design LRU Cache. [Goldman Sachs] [AppDynamics]

8. Design a JSON Parser. [ServiceNow]
9. Design a Vending Machine. [Amazon]
10. Design a Food Delivery System. [Amazon]
11. Design a Calendar like application (Google Calendar). [Flipkart]
12. Design BlackJack. [Gojek]
13. Design a Web Browser. [Bloomberg]
14. Design a Parking Lot. [Google] [Amazon]