## **Basic OOPs Interview Questions**

## **1. What is OOPs?**

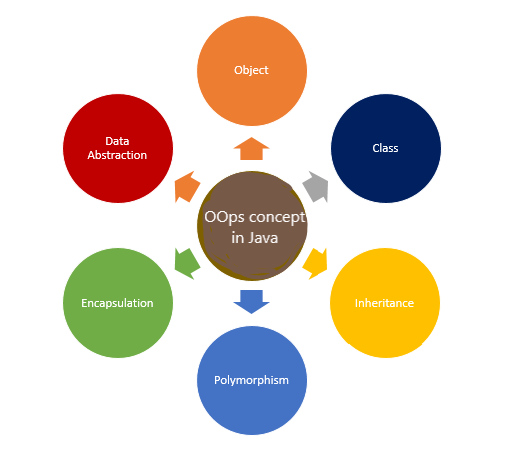
Programmers can use objects to represent real-world circumstances thanks to object-oriented programming. Any entity with states and behaviors is an object. While methods define an item’s behaviors, states reflect the characteristics or data of an entity. Objects include students, workers, books, etc. By exchanging messages, these things communicate with one another. A class is also a template for building an object. A class is required in order to generate objects. For instance, there needs to be an Employee class in order to generate an Employee object.

### **2. Difference between Procedural programming and OOPs?**

| **Procedural Programming** | **Oops** |
| --- | --- |
| Procedural Programming is based on functions. | Object-oriented programming is based on real-world objects. |
| It shows the data to the entire program. | It encapsulates the data. |
| It does not have a scope for code reuse. | It provides more scope for code reuse. |
| It follows the concept of top-down programming. | It follows a bottom-up programming paradigm. |
| The nature of the language is complicated. | It is less complicated in nature, so it is easier to modify, extend and maintain. |
| It is hard to modify, extend and maintain the code. |  |

### **3. Why use OOPs?**

Programming with OOP lets you package together data states and functionality to change those data states while keeping the specifics secret (Consider the analogy of a car, you can only see the steering of the car while driving, the circuitry behind it is hidden from you). As a result, OOP design produces flexible, modular, and abstract code. Because of this, it is very helpful for developing larger programs. Using classes and objects, you may include OOP into your code. The objects you construct will have the states and capabilities of the class to which they belong.

[](https://www.mygreatlearning.com/academy/learn-for-free/courses/oops-in-python?gl_blog_id=85199)

### **[OOPs in Java](https://www.mygreatlearning.com/academy/learn-for-free/courses/oops-in-java?gl_blog_id=85199)**

[Gain a competitive edge](https://www.mygreatlearning.com/academy/learn-for-free/courses/oops-in-java?gl_blog_id=85199)

[](https://www.mygreatlearning.com/academy/learn-for-free/courses/oops-in-java?gl_blog_id=85199)

### **[OOPs in C++](https://www.mygreatlearning.com/academy/learn-for-free/courses/oops-concepts-in-c?gl_blog_id=85199)**

[Land your dream job](https://www.mygreatlearning.com/academy/learn-for-free/courses/oops-concepts-in-c?gl_blog_id=85199)

### **4. What are the basic concepts of OOPs?**

The basic concepts of OOPs are:

* Inheritance
* Encapsulation
* Polymorphism
* Abstraction

### **5. What is Encapsulation?**

Encapsulation is also a part of the 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 real-world objects. It “shows” only essential attributes and “hides” unnecessary information from the outside. The main focus of abstraction is to hide 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, or both. These methods are known as overloaded methods and this feature is called [**method overloading**](https://www.mygreatlearning.com/blog/method-overloading-in-java/).

### **8. What is method overriding?**

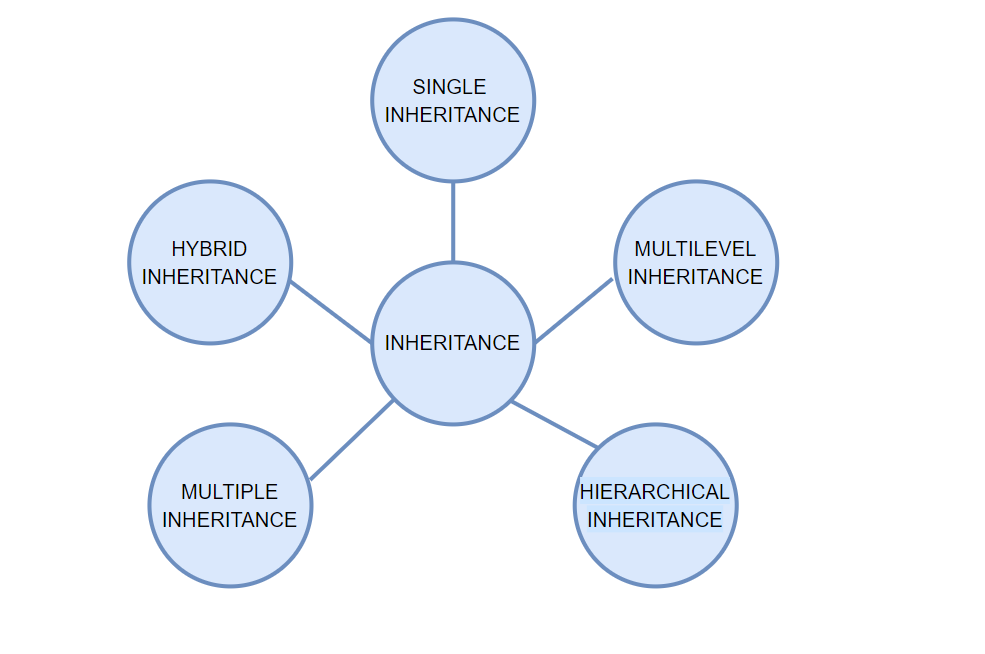
**Method overriding** is a concept of object-oriented programming.

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 superclasses or parent classes.

### **9. Types of Inheritance in OOPS**

Different types of inheritances in OOps are as follows:

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



### **10. What are the main features of OOPs?**

The main features of OOPs are given as follows:

* In OOP, you combine the code into one unit so you can specify the parameters of each piece of data. This process of wrapping up data into a single unit is called encapsulation.
* By using classes, you can generalise your object types and make your application easier to use. This is termed as an abstraction.
* The ability for a class to inherit characteristics and behaviours from another class allows for more code reuse.
* Polymorphism allows for the creation of several objects from a single, adaptable class of code.

### **11. Is it possible to call the base class method without creating an instance?**

Yes, we can possibly call the base class method without creating an instance in the following 3 cases:

1. If the method is static
2. Calling the inherited method inside a derived class
3. Calling the method using the base keyword from the sub-classes

The most popular case is that of the static methods.

### **12. What are the limitations of OOPs?**

Following are some of the common limitations of OOPs:

* Size exceeds that of other programs.
* It took a lot of work to make, and it runs more slowly than other programs.
* It is inappropriate for certain types of issues.
* It takes some getting used to.

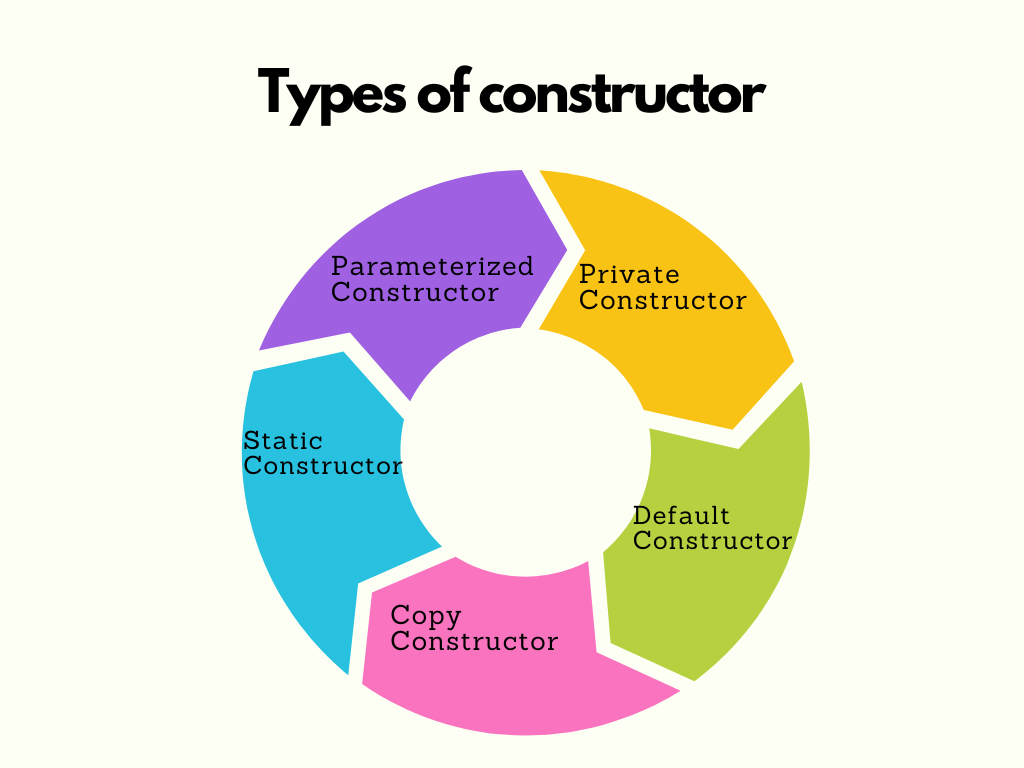
### **13. What are constructors?**

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 constructors 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.

A user-defined class serves layout or blueprint from which objects can be built. In essence, a class is made up of fields known as attributes and methods known as member functions that define actions. A structure is a grouping of variables of various data kinds under one heading.

### **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 languages come under the oops concept?**

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

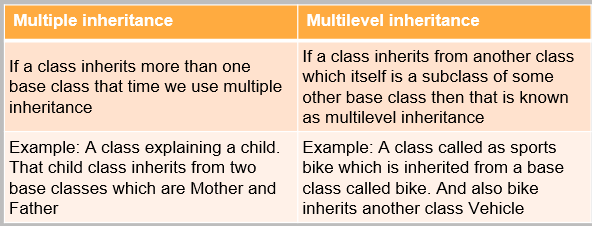
* Java
* JavaScript
* Python
* C++
* Visual Basic
* .NET
* Ruby
* Scala
* PHP

Check out the OOPs concept in Python Video.

### **18. What is inheritance?**

Whenever one class is derived from another, it is referred to as inheritance. The child class will inherit all of the parent class’s public and protected properties and methods. Apart from the attributes and methods inherited from the parent class, it can also have its own additional set of features. The’ extends’ keyword is used to specify an inherited class.

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.



### **19. What is hybrid inheritance?**

The type of inheritance formed by the combination of different types of inheritances like single, multiple inheritances, etc. is classified as hybrid inheritance.

### **20. What is hierarchical inheritance?**

In the case of a hierarchical inheritance, multiple subclasses inherit from a parent class. Hierarchical inheritance is a type of inheritance in which multiple classes are descended from a single parent or base class. 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 is a class from which a subclass or child class is derived. Base class and parent class are other names for a superclass. For example, if Student is a class derived from the Person class, then the Person class will be referred to as the 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 class that derives from another class is referred to as a subclass. A subclass inherits the properties of its ancestors or parent classes. For example, the class Bike is a subclass or a derivative of the Vehicle class.

### **24. What is Polymorphism?**

**Polymorphism** is one of the most used and 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?**

In OOP, static polymorphism determines which method to call at compile time. For the same trigger with static polymorphism, the object might respond differently. Function, constructor and operator overloading are examples of static polymorphism.

### **26. What is dynamic polymorphism?**

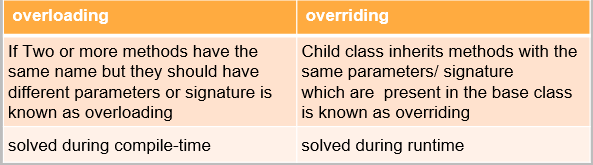
Dynamic polymorphism is a method or process that handles a call to an overridden method during runtime rather than at compile time. It is also referred to as dynamic method dispatch or runtime polymorphism. Using method overriding, we can create dynamic polymorphism. An example of runtime polymorphism: is method overriding.

### **27. What is operator overloading?**

The user-defined data type is given a special meaning by the operator using [operator overloading](https://www.mygreatlearning.com/blog/operator-overloading-in-cpp/). It is a compile-time polymorphism.

### **28. Differentiate between overloading and overriding.**

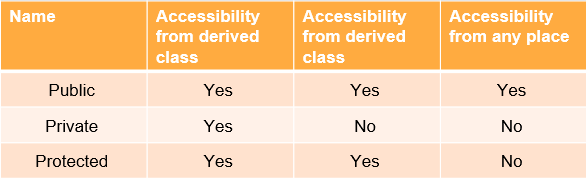
When two or more methods in the same class have the same name but different parameters, this is referred to as overloading. The technique of using the same method signature, i.e., name and parameters, in both the superclass and the child class is known as overriding.



### **29. What is encapsulation?**

Encapsulation is used to wrap the data and the code which works 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?**

****

### **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

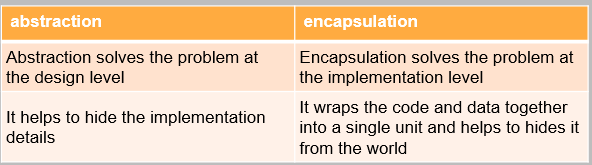
### **33. What is an abstract class?**

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

**So what is an 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 have to be exclusively defined in the subclass.

### **34. Differentiate between data abstraction and encapsulation.**

****

### **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 that 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?**

By copying the members of an existing object, the copy constructor initialises the members of a newly formed object. The argument for the copy constructor is a reference to an object of the same class. Programmers have the option of directly defining the [copy constructor](https://www.mygreatlearning.com/blog/copy-constructor-in-cpp/). The compiler defines the copy constructor if the programmer doesn’t.

### **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)?**

Programming languages like C# and Java include garbage collection (GC) as a memory recovery mechanism. A programming language that supports garbage collection (GC) contains one or more GC engines that automatically release memory space that has been reserved for things the application is no longer using.

### **40. What is a final variable?**

A final variable can only receive one explicit initialization. A reference variable that has been marked as final is unchangeable in its object reference. The data included in the object, however, can be modified. As a result, while the object’s state can be altered, its reference cannot.

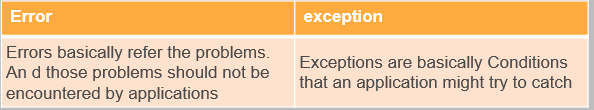
### **41. What is an exception?**

An exception is a kind of message that interrupts and comes up when there is an issue with the normal execution of a program. Exceptions provide an error and transfer it 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?**

****

### **44. What is a try/ catch block?**

The terms “try” and “catch” describe how to handle exceptions brought on by coding or data mistakes while a program is running. The section of code where exceptions occur is called a try block. Exceptions from try blocks are caught and handled in a catch block.

### **45. What is a finally block?**

Finally designates the section of code that works with the try keyword. It specifies code that is always executed before the method is finished, immediately behind the try and any catch blocks. Regardless of whether an exception is thrown or caught, the finally block is always executed.

### **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?**

The key distinction between structured and object-oriented programming is that the former allows for the creation of programs using a collection of modules or functions, whilst the latter allows for the construction of programs using a collection of objects and their interactions.

Object-oriented programming involves concepts of objects and classes. Everything is considered as an object which has specific properties and behaviours 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 behaviours, and the similar type of objects having similar features and behaviours 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 behaviour 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 the 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.

## **OOPs Interview Questions for Experienced**

### **51. What is an interface?**

An interface is a user-defined [data type](https://www.mygreatlearning.com/blog/types-of-data/) 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 behaviours, and an interface contains behaviours 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 behaviour of the objects.

Methods are programming constructs that perform specific tasks/behaviour.

### **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 an abstract class should be used. If the implementation is known completely, then a concrete Class should be used.

### **55. What are the limitations of OOPs?**

1. Larger Program size – Programs can become lengthy if written using [OOps concepts](https://www.mygreatlearning.com/blog/oops-concepts-in-java/) 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 a 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 are 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 them in Java programming. Variable initialization can be static or dynamic. The syntax for variable declaration and static initialization is: –

***Types of variables***

* **Primitive Variables:** It is used to represent primitive values like int, float, etc.
* Reference Variables: It is used to refer to objects in Java.
* **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
* **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.
* **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.

### **63. What do you understand by Garbage Collection in the OOPs world?**

Garbage collection is a memory recovery technique included in programming languages like C# and Java. A GC-enabled programming language contains one or more garbage collectors that automatically free up memory space allocated to objects that are no longer needed by the program.

### **64. Is it possible to run a Java application without implementing the OOPs concept?**

No, since Java programmes are founded on the concept of object-oriented programming models, or OOPs, a Java application cannot be implemented without it.

### **65. What is the output of the below code?**

| *class Person*  *{*  *private String show()*  *{*  *return “This is a person”;*  *}*  *}*  *class Teacher extends Person*  *{*  *protected String show()*  *{*  *return “This is a teacher”;*  *}*  *}*  *public class MathsTeacher extends Person*  *{*  *@Override public final String show()*  *{*  *return “This is a Maths teacher”;*  *}*  *public static void main(String[] name)*  *{*  *final Person mt = new MathsTeacher();*  *System.out.print(mt.show());*  *}*  *}* |
| --- |

The output will be: This is a Maths teacher

### **66. Find the output of the below code.**

| *class Arithmetic*  *{*  *public final double var = 5;*  *}*  *class DeepArith extends Arithmetic*  *{*  *public final double var = 10;*  *}*  *public class AdvancedArith extends DeepArith*  *{*  *public final double secret = 20;*  *public static void main(String[] num)*  *{*  *Arithmetic arith = new AdvancedArith();*  *System.out.print(arith.var);*  *}*  *}* |
| --- |

The correct output for this code is 5.

### **67. Predict the output of the following.**

| *class Parent*  *{*  *public void display()*  *{*  *System.out.println(“Parent”);*  *}*  *}*  *class Child extends Parent*  *{ private void display()*  *{ System.out.println(“Child”);*  *}*  *}*  *public class main*  *{*  *public static void main(String args[])*  *{*  *Parent node = new Child(); node.show();*  *}*  *}* |
| --- |

Running this code will generate a compile error as a sub-class function overriding a super class function cannot be given more restrictive access.

### **68. Implement a Singleton class in Java.**

public class Singleton {

private static Singleton instance;

private Singleton() {

// Private constructor to prevent instantiation.

}

public static Singleton getInstance() {

if (instance == null) {

synchronized (Singleton.class) {

if (instance == null) {

instance = new Singleton();

}

}

}

return instance;

}

}

### **69. Implement a Stack data structure using a LinkedList in Python.**

class Node:

def \_\_init\_\_(self, data=None):

self.data = data

self.next = None

class Stack:

def \_\_init\_\_(self):

self.head = None

def is\_empty(self):

return self.head is None

def push(self, data):

new\_node = Node(data)

new\_node.next = self.head

self.head = new\_node

def pop(self):

if self.is\_empty():

raise Exception("Stack is empty")

popped\_value = self.head.data

self.head = self.head.next

return popped\_value

### **70. Implement an abstract class in C# with an abstract method.**

abstract class Shape

{

public abstract double CalculateArea();

}

class Circle : Shape

{

private double radius;

public Circle(double radius)

{

this.radius = radius;

}

public override double CalculateArea()

{

return Math.PI \* radius \* radius;

}

}

Check out OOPs in Java Video



## **Frequently Asked OOPs Interview Questions**

**Q: What are the 4 basics of OOP?**

**A:** OOP stands for Object-Oriented Programming, and its four basic principles are Encapsulation, Abstraction, Polymorphism, and Inheritance. OOP enables programmers to consider software development as if they are working with actual entities. In OOP, some objects have a field where data/knowledge can be stored and can do several methods.

**Q: What is the object-oriented programming interview?**

**A:** Object-Oriented Programming, also usually called OOPS, is a kind of programming that is more object-based and not just based on functions or procedures. Individual objects are collected into several classes. Real-world entities such as inheritance, polymorphism, and hiding are implemented by OOPS into programming. It also enables binding data as well as code together.

**Q: What are the 3 principles of OOP?**

**A:** The three main principles of Object-Oriented Programming are Encapsulation, inheritance, and polymorphism.

**Q: What is the concept of OOPS?**

**A:** OOPS or Object-Oriented Programming System is a programming concept that mainly works based on Encapsulation, Abstraction, Polymorphism, and Inheritance. The usual concept of OOPs is to create objects, use them again all through the program, and finally manipulate these objects to fetch our results.

**Q: Why is OOPS used?**

**A:** The main aim of an Object-Oriented Programming System is to implement real-world entities such as polymorphism, inheritance, hiding, and many more in programming. The aim lies in binding together the data as well as functions that work on them so that other parts of the code cannot access the data other than that function.

**Q: What is polymorphism in OOPS?**

**A:** Polymorphism in an Object-Oriented Programming System is a feature of object-based programming languages that enable a particular routine to use variables of several types at different times. It can also be called the ability of a programming language to present the same interface for different primary data types.

**Q: Who is the father of OOPS?**

**A:** The father of the Object-Oriented Programming System is considered to be Alan Kay by some people. He identified some characteristics as basics to OOP Kay 1993:1. He coined OOPs around 1966 or 1967 when he was at grad school.

**Q: What are the main features of OOPS?**

**A:** Some of the main features in OOPS include Classes, Objects, Data Abstraction, Encapsulation, Inheritance, and Polymorphism. OOP is a programming paradigm that is based on the idea of objects.

**Q: What are the advantages of OOPS?**

**A:** Since OOP is one of the main development approaches which is easily accepted, the advantages are many. Some of the advantages of OOPS include Reusability, Data Redundancy, Code Maintenance, Security, Design Benefits, Easy Troubleshooting, Better Productivity, Polymorphism Flexibility, and Problem-solving.