OOPs Interview Questions

Object-oriented programming (OOPs) is a programming paradigm that is based on the concept of objects rather than just functions and procedures. It is the most popular methodology among developers.

Nowadays tech giants demanding and hiring who has expertise in **object-oriented** approaches and patterns and conducting interviews for the same. The advantage of hiring such candidates is that they can also learn other OOP languages easily as per organization requirements. Since, going through the section, you can increase your chance to get hire by companies if you have well prepared for **OOPs interview questions**.

In this section, we have collected some commonly asked **OOPs interview questions** for both fresher and experienced. It can help you to crack the interview to get your dream job.

1) What do you understand by OOP?

OOP stands for object-oriented programming. It is a programming paradigm that revolves around the object rather than function and procedure. In other words, it is an approach for developing applications that emphasize on objects. An object is a real word entity that contains data and code. It allows binding data and code together.

---

2) Name any seven widely used OOP languages.

There are various OOP languages but the most widely used are:

- Python
- Java
- Go
- Dart
- C++
- C#
- Ruby

---

3) What is the purpose of using OOPs concepts?

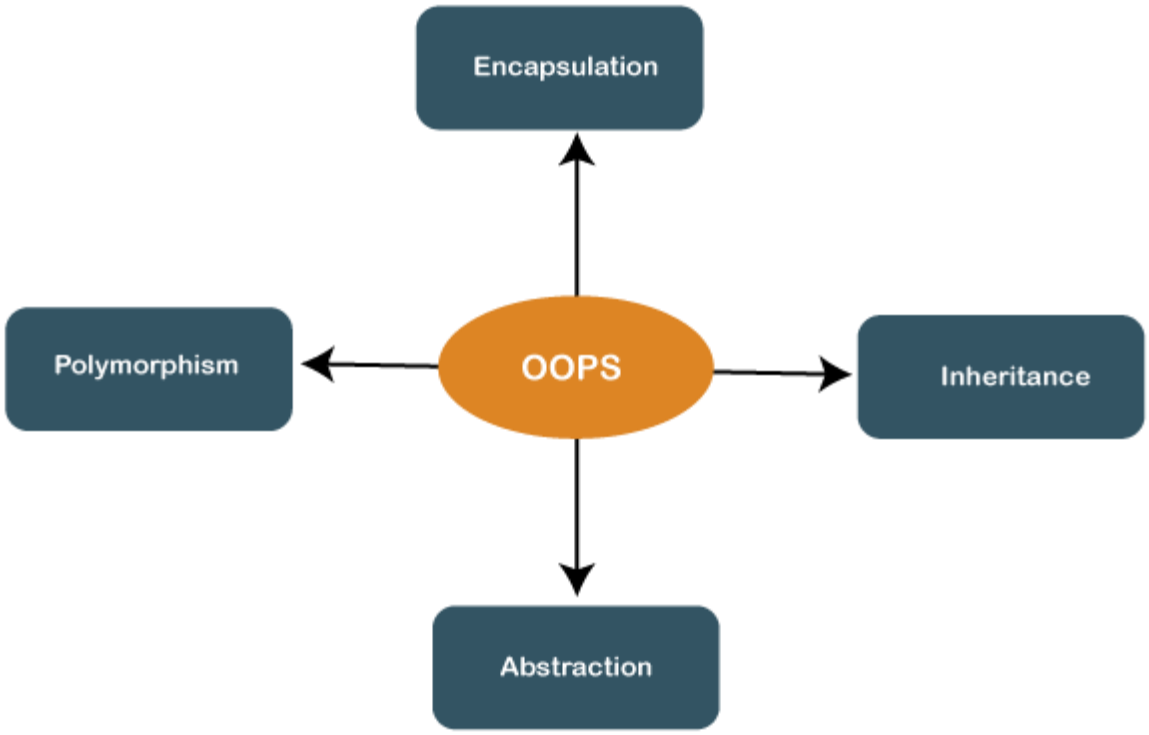
The aim of OOP is to implement real-world entities like inheritance, hiding, polymorphism in programming. The main purpose of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

---

4) What are the four main features of OOPs?

The OOP has the following four features:

- Inheritance
- Encapsulation
- Polymorphism
- Data Abstraction



---

5) Why OOP is 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.

---

6) What are the advantages and disadvantages of OOP?

Advantages of OOP

- It follows a bottom-up approach.
- It models the real word well.
- It allows us the reusability of code.
- Avoids unnecessary data exposure to the user by using the abstraction.
- OOP forces the designers to have a long and extensive design phase that results in better design and fewer flaws.
- Decompose a complex problem into smaller chunks.

- Programmer are able to reach their goals faster.
- Minimizes the complexity.
- Easy redesign and extension of code that does not affect the other functionality.

Disadvantages of OOP

- Proper planning is required.
- Program design is tricky.
- Programmer should be well skilled.
- Classes tend to be overly generalized.

---

7) What are the limitations of OOPs?

- Requires intensive testing processes.
- Solving problems takes more time as compared to Procedure Oriented Programming.
- The size of the programs created using this approach may become larger than the programs written using the procedure-oriented programming approach.
- Software developed using this approach requires a substantial amount of pre-work and planning.
- OOP code is difficult to understand if you do not have the corresponding class documentation.
- In certain scenarios, these programs can consume a large amount of memory.
- Not suitable for small problems.
- Takes more time to solve problems.

---

8) What are the differences between object-oriented programming and structural programming?

Object-oriented Programming	Structural Programming
It follows a bottom-up approach.	It follows a top-down approach.

It provides data hiding.	Data hiding is not allowed.
It is used to solve complex problems.	It is used to solve moderate problems.
It allows reusability of code that reduces redundancy of code.	Reusability of code is not allowed.
It is based on objects rather than functions and procedures.	It provides a logical structure to a program in which the program is divided into functions.
It provides more security as it has a data hiding feature.	It provides less security as it does not support the data hiding feature.
More abstraction more flexibility.	Less abstraction less flexibility.
It focuses on data.	It focuses on the process or logical structure.

---

9) What do you understand by pure object-oriented language? Why Java is not a pure object-oriented programming language?

The programming language is called pure object-oriented language that treats everything inside the program as an object. The primitive types are not supported by the pure OOPs language. There are some other features that must satisfy by a pure object-oriented language:

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction
- All predefined types are objects
- All user-defined types are objects
- All operations performed on objects must be only through methods exposed to the objects.

Java is not a pure object-oriented programming language because pre-defined data types in Java are not treated as objects. Hence, it is not an object-oriented language.

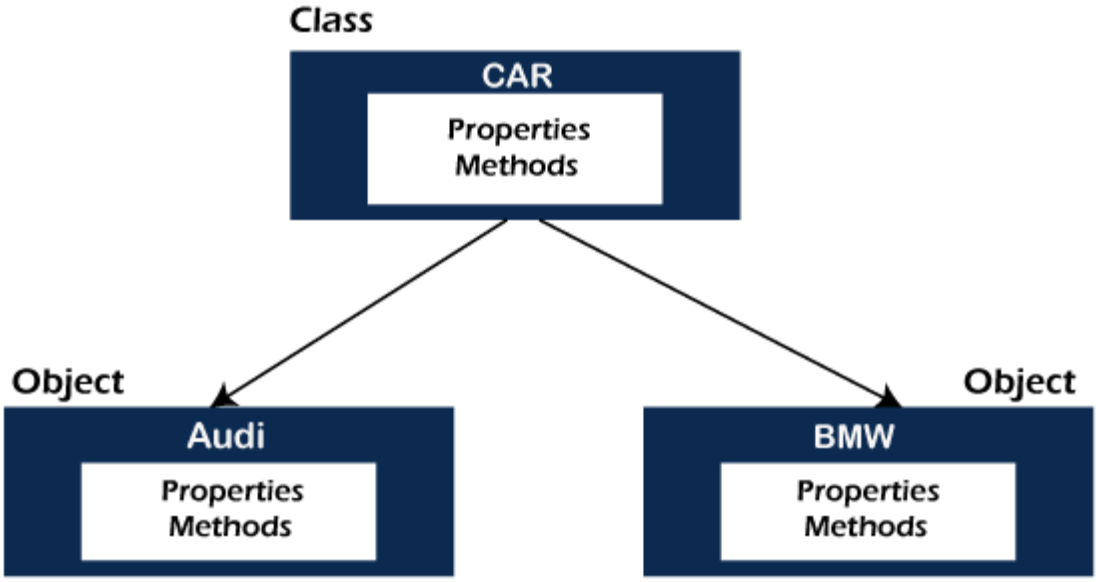
---

10) What do you understand by class and object? Also, give example.

Class: A class is a blueprint or template of an object. It is a user-defined data type. Inside a class, we define variables, constants, member functions, and other functionality. It does not consume memory at run time. Note that classes are not considered as a data structure. It is a logical entity. It is the best example of data binding.

Object: An object is a real-world entity that has attributes, behavior, and properties. It is referred to as an instance of the class. It contains member functions, variables that we have defined in the class. It occupies space in the memory. Different objects have different states or attributes, and behaviors.

The following figure best illustrates the class and object.



Class	Object
It is a logical entity.	It is a real-world entity.
It is conceptual.	It is real.
It binds data and methods together into a single unit.	It is just like a variable of a class.
It does not occupy space in the memory.	It occupies space in the memory.
It is a data type that represents the blueprint of an object.	It is an instance of the class.
It is declared once.	Multiple objects can be declared as and when required.
It uses the keyword class when declared.	It uses the new keyword to create an object.
A class can exist without any object.	Objects cannot exist without a class.

11) What are the differences between class and object?

---

12) What are the key differences between class and structure?

Class	Structure
Class is a group of common objects that shares common properties.	The structure is a collection of different data types.
It deals with data members and member functions.	It deals with data members only.
It supports inheritance.	It does not support inheritance.
Member variables cannot be initialized directly.	Member variables can be initialized directly.
It is of type reference.	It is of a type value.
It's members are private by default.	It's members are public by default.
The keyword class defines a class.	The keyword struct defines a structure.
An instance of a class is an object.	An instance of a structure is a structure variable.

Useful while dealing with the complex data structure.	Useful while dealing with the small data structure.

13) What is the concept of access specifiers when should we use these?

In OOPs language, **access specifiers** are reserved keyword that is used to set the accessibility of the classes, methods and other members of the class. It is also known as **access modifiers**. It includes **public, private,** and **protected**. There is some other access specifier that is language-specific. Such as Java has another access specifier **default**. These access specifiers play a vital role in achieving one of the major functions of OOP, i.e. encapsulation. The following table depicts the accessibility.

Specifiers	Within Same Class	In Derived Class	Outside the Class
Private	Yes	No	No
Protected	Yes	Yes	No
Public	Yes	Yes	Yes

14) What are the manipulators in OOP and how it works?

Manipulators are helping functions. It is used to manipulate or modify the input or output stream. The modification is possible by using the **insertion** (<<) and **extraction** (>>) operators. Note that the modification of input or output stream does not mean to change the values of variables. There are two types of manipulators with **arguments** or **without arguments.**

The example of manipulators that do not have arguments is **endl, ws, flush,** etc. Manipulators with arguments are **setw(val), setfill(c), setbase(val), setiosflags(flag).** Some other manipulators are **showpos, fixed, scientific, hex, dec, oct,** etc.

15) What are the rules for creating a constructor?

- It cannot have a return type.
- It must have the same name as the Class name.
- It cannot be marked as static.
- It cannot be marked as abstract.
- It cannot be overridden.
- It cannot be final.

16) What are the differences between the constructor and the method in Java?

Constructor	Method
Constructor has the same name as the class name.	The method name and class name are not the same.
It is a special type of method that is used to initialize an object of its class.	It is a set of instructions that can be invoked at any point in a program.
It creates an instance of a class.	It is used to execute Java code.
It is invoked implicitly when we create an object of the class.	It gets executed when we explicitly called it.
It cannot be inherited by the subclass.	It can be inherited by the subclass.
It does not have any return type.	It must have a return type.
It cannot be overridden in Java.	It can be overridden in Java.
It cannot be declared as static.	It can be declared as static.
Java compiler automatically provides a default constructor.	Java compiler does not provide any method by default.

17) How does procedural programming be different from OOP

Procedural Oriented Programming	Object-Oriented Programming
It is based on functions.	It is based on real-world objects.
It follows a top-down approach.	It follows a bottom-up approach.
It is less secure because there is no proper way to hide data.	It provides more security.
Data is visible to the whole program.	It encapsulates the data.
Reuse of code is not allowed.	The code can be reused.
Modification and extension of code are not easy.	We can easily modify and extend code.
Examples of POP are C, VB, FORTRAN, Pascal, etc.	Examples of OOPs are C++, Java, C#, .NET, etc.

differ?

---

18) What are the differences between error and exception?

Basis of Comparison	Exception	Error
Recoverable/ Irrecoverable	Exception can be recovered by using the try-catch block.	An error cannot be recovered.
Type	It can be classified into two categories i.e. checked and unchecked.	All errors in Java are unchecked.
Occurrence	It occurs at compile time or run time.	It occurs at run time.
Package	It belongs to java.lang.Exception package.	It belongs to java.lang.Error package.
Known or unknown	Only checked exceptions are known to the compiler.	Errors will not be known to the compiler.

Causes	It is mainly caused by the application itself.	It is mostly caused by the environment in which the application is running.
Example	**Checked Exceptions:** SQLException, IOException **Unchecked Exceptions:** ArrayIndexOutOfBoundException, NullPointerException, ArithmaticException	Java.lang.StackOverFlow, java.lang.OutOfMemoryError

---

19) What are the characteristics of an abstract class?

An abstract class is a class that is declared as abstract. It cannot be instantiated and is always used as a base class. The characteristics of an abstract class are as follows:

- Instantiation of an abstract class is not allowed. It must be inherited.
- An abstract class can have both abstract and non-abstract methods.
- An abstract class must have at least one abstract method.
- You must declare at least one abstract method in the abstract class.
- It is always public.
- It is declared using the **abstract**

The purpose of an abstract class is to provide a common definition of the base class that multiple derived classes can share.

---

20) Is it possible for a class to inherit the constructor of its base class?

No, a class cannot inherit the constructor of its base class.

---

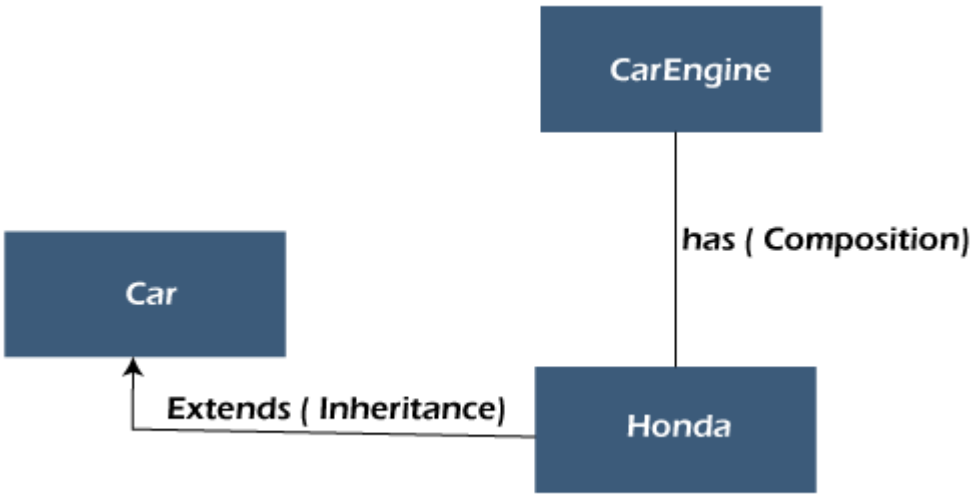
21) Identify which OOPs concept should be used in the following scenario?

A group of 5 friends, one boy never gives any contribution when the group goes for the outing. Suddenly a beautiful girl joins the same group. The boy who never contributes is now spending a lot of money for the group.

Runtime Polymorphism

22) What is composition?

Composition is one of the vital concepts in OOP. It describes a class that references one or more objects of other classes in instance variables. It allows us to model a has-a association between objects. We can find such relationships in the real world. For example, a car has an engine. the following figure depicts the same



The main benefits of composition are:

- Reuse existing code
- Design clean APIs
- Change the implementation of a class used in a composition without adapting any external clients.

23) What are the differences between copy constructor and assignment operator?

The copy constructor and the assignment operator (=) both are used to initialize one object using another object. The main difference between the two is that the copy constructor allocates separate memory to both objects i.e. existing object and newly created object while the assignment operator does not allocate new memory for the newly created object. It uses the reference variable that points to the previous memory block (where an old object is located).

Syntax of Copy Constructor

```
1. class_name (const class_name &obj)
2. {
3. //body
4. }
```

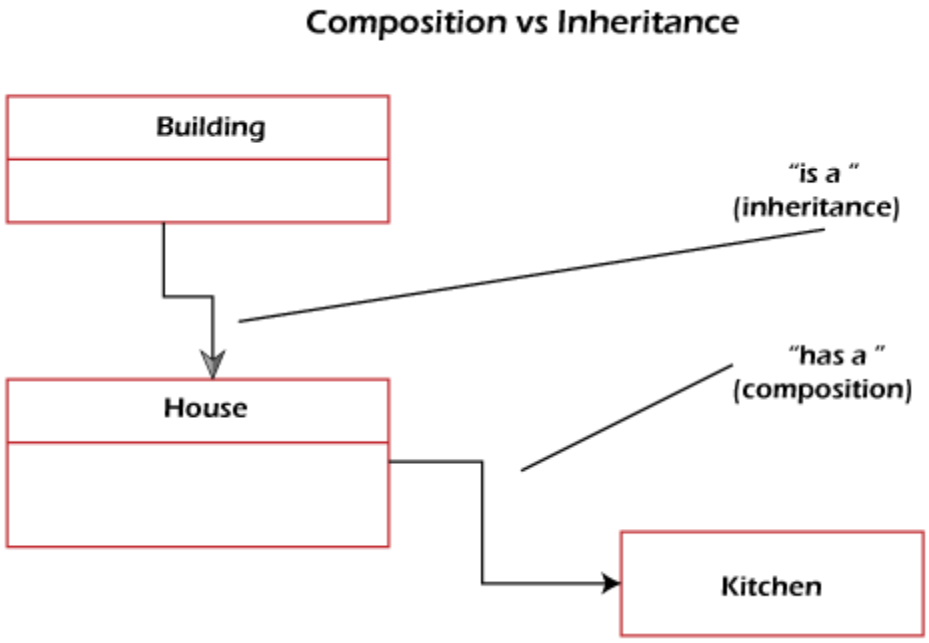
Syntax of Assignment Operator

```
1. class_name obj1, obj2;
2. obj1=obj2;
```

Copy Constructor	Assignment Operator
It is an overloaded constructor.	It is an operator.
It creates a new object as a copy of an existing object.	It assigns the value of one object to another object both of which already exist.
The copy constructor is used when a new object is created with some existing object.	It is used when we want to assign an existing object to a new object.
Both the objects use separate memory locations.	Both objects share the same memory but use the two different reference variables that point to the same location.
If no copy constructor is defined in the class, the compiler provides one.	If the assignment operator is not overloaded then the bitwise copy will be made.

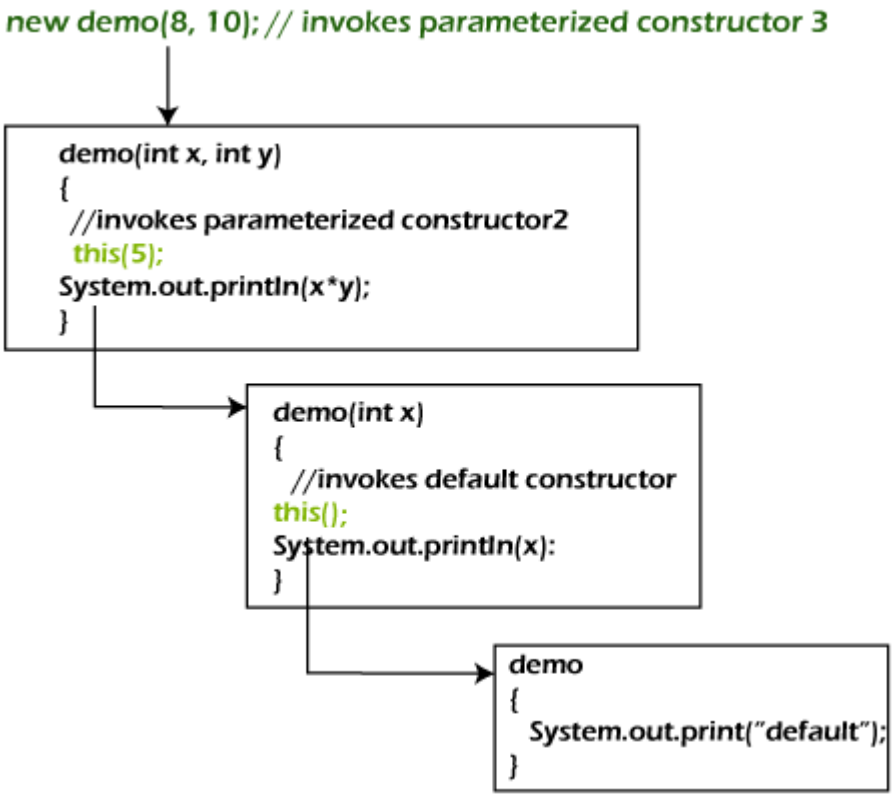
24) What is the difference between Composition and Inheritance?

Inheritance means an object inheriting reusable properties of the base class. Compositions mean that an object holds other objects. In Inheritance, there is only one object in memory (derived object) whereas, in Composition, the parent object holds references of all composed objects. From a design perspective, inheritance is "is a" relationship among objects whereas Composition is "has a" relationship among objects.



25) What is constructor chaining?

In OOPs, constructor chaining is a sequence of invoking constructors (of the same class) upon initializing an object. It is used when we want to invoke a number of constructors, one after another by using only an instance. In other words, if a class has more than one constructor (overloaded) and one of them tries to invoke another constructor, this process is known as constructor chaining. In C++, it is known as constructor delegation and it is present from C++ 11.



---

26) What are the limitations of inheritance?

- The main disadvantage of using inheritance is two classes get tightly coupled. That means one cannot be used independently of the other. If a method or aggregate is deleted in the Super Class, we have to refactor using that method in SubClass.
- Inherited functions work slower compared to normal functions.
- Need careful implementation otherwise leads to improper solutions.

---

27) What are the differences between Inheritance and Polymorphism?

Inheritance	Polymorphism
Inheritance is one in which a derived class inherits the already existing class's features.	Polymorphism is one that you can define in different forms.
It refers to using the structure and behavior of a superclass in a subclass.	It refers to changing the behavior of a superclass in the subclass.
It is required in order to achieve polymorphism.	In order to achieve polymorphism, inherence is not required.
It is applied to classes.	It is applied to functions and methods.
It can be single, hybrid, multiple, hierarchical, multipath, and multilevel inheritance.	There are two types of polymorphism compile time and run time.
It supports code reusability and reduces lines of code.	It allows the object to decide which form of the function to be invoked at run-time (overriding) and compile-time (overloading).

28) What is Coupling in OOP and why it is helpful?

In programming, separation of concerns is known as **coupling**. It means that an object cannot directly change or modify the state or behavior of other objects. It defines how closely two objects are connected together. There are two types of coupling, **loose** coupling, and **tight** coupling.

Objects that are independent of one another and do not directly modify the state of other objects is called loosely coupled. Loose coupling makes the code more flexible, changeable, and easier to work with.

Objects that depend on other objects and can modify the states of other objects are called tightly coupled. It creates conditions where modifying the code of one object also requires changing the code of other objects. The reuse of code is difficult in tight coupling because we cannot separate the code.

Since using loose coupling is always a good habit.

29) Name the operators that cannot be overload.

1. Scope Resolution Operator (::)
2. Ternary Operator (? :)
3. Member Access or Dot Operator (.)
4. Pointer to Member Operator (.*)
5. sizeof operator

---

30) 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.

virtual: indicates that a method may be overridden by an inheritor

override: Overrides the functionality of a virtual method in a base class, providing different functionality.

new: Hides the original method (which doesn't have to be virtual), providing different functionality. This should only be used where it is absolutely necessary.

When you hide a method, you can still access the original method by upcasting to the base class. This is useful in some scenarios, but dangerous.

---

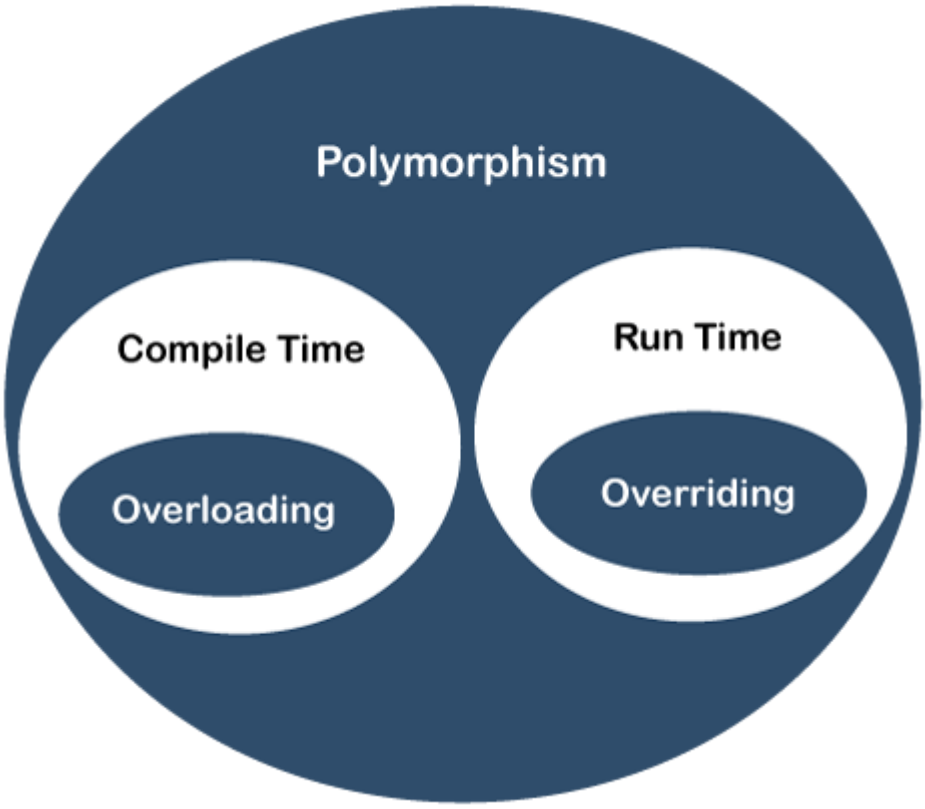
31) Explain overloading and overriding with example?

Overloading

Overloading is a concept in OOP when two or more methods in a class with the same name but the method signature is different. It is also known as **compile-time polymorphism**. For example, in the following code snippet, the method **add()** is an overloaded method.

```
1. public class Sum
2. {
3. int a, b, c;
```

```
4.  public int add();
5.  {
6.  c=a+b;
7.  return c;
8.  }
9.  add(int a, int b);
10. {
11. //logic
12. }
13. add(int a, int b, int c);
14. {
15. //logic
16. }
17. add(double a, double b, double c);
18. {
19. //logic
20. }
21. //statements
22. }
```



Overriding

If a method with the same method signature is presented in both child and parent class is known as method **overriding**. The methods must have the same number of parameters and the same type of parameter. It overrides the value of the parent class method. It is also known as **runtime polymorphism**. For example, consider the following program.

```
1. class Dog
2. {
3. public void bark()
4. {
5. System.out.println("woof ");
6. }
7. }
8. class Hound extends Dog
9. {
10. public void sniff()
11. {
```

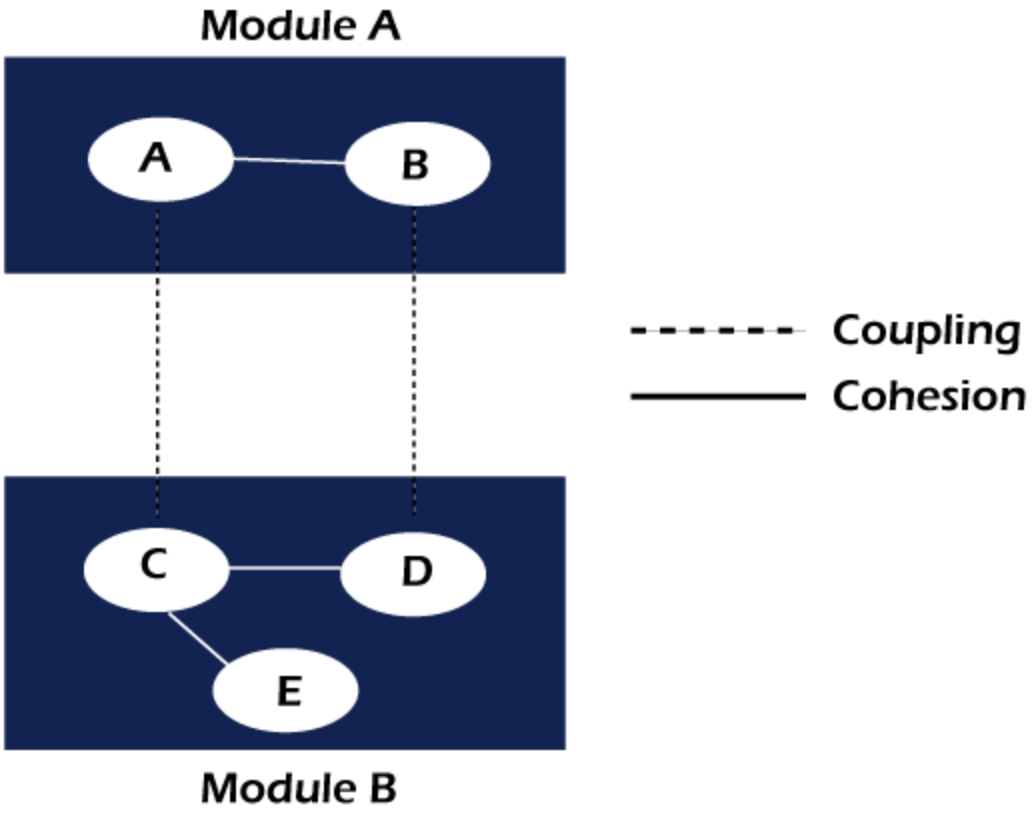
```
12. System.out.println("sniff ");
13. }
14. //overrides the method bark() of the Dog class
15. public void bark()
16. {
17. System.out.println("bowl");
18. }
19. }
20. public class OverridingExample
21. {
22. public static void main(String args[])
23. {
24. Dog dog = new Hound();
25. //invokes the bark() method of the Hound class
26. dog.bark();
27. }
28. }
```

---

32) What is Cohesion in OOP?

In OOP, **cohesion** refers to the degree to which the elements inside a module belong together. It measures the strength of the relationship between the module and data. In short, cohesion represents the clarity of the responsibilities of a module. It is often contrasted with coupling.

It focuses on a how single module or class is intended. Higher the cohesiveness of the module or class, better is the object-oriented design.



There are two types of cohesion, i.e. **High** and **Low**.

- High cohesion is associated with several required qualities of software including **robustness, reliability**, and **understandability**.
- Low cohesion is associated with unwanted qualities such as being difficult to **maintain, test, reuse,** or even **understand**.

High cohesion often associates with loose coupling and vice versa.

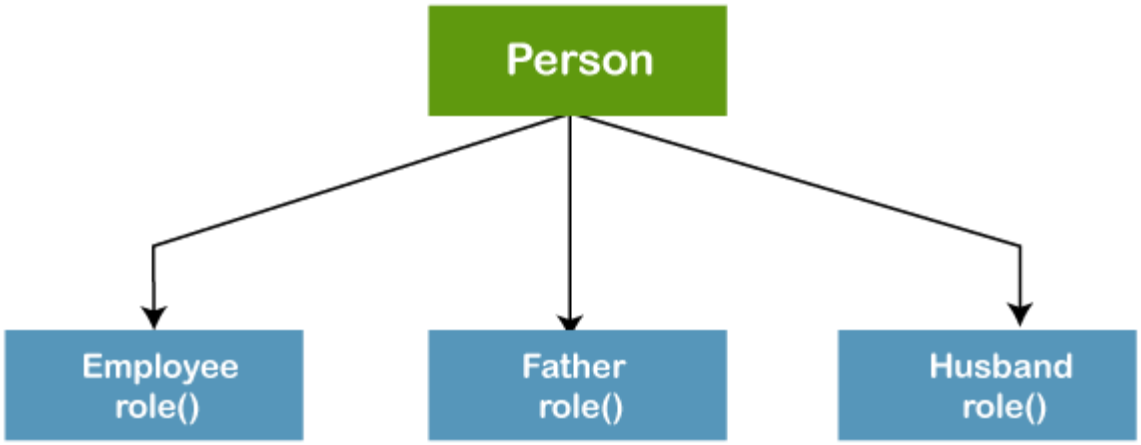
---

33) Give a real-world example of polymorphism?

The general meaning of Polymorphism is one that has different forms. The best real-world example of polymorphism is a **person** that plays different roles at different palaces or situations.

- At home a person can play the role of father, husband, and son.
- At the office the same person plays the role of boss or employee.
- In public transport, he plays the role of passenger.
- In the hospital, he can play the role of doctor or patient.

- At the shop, he plays the role of customer.



Hence, the same person possesses different behavior in different situations. It is called polymorphism.

---

34) 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.

---

35) What is data abstraction and how can we achieve data abstraction?

It is one of the most important features of OOP. It allows us to show only essential data or information to the user and hides the implementation details from the user. A real-world example of abstraction is driving a car. When we drive a car, we do not need to know how the engine works (implementation) we only know how ECG works.

There are two ways to achieve data abstraction

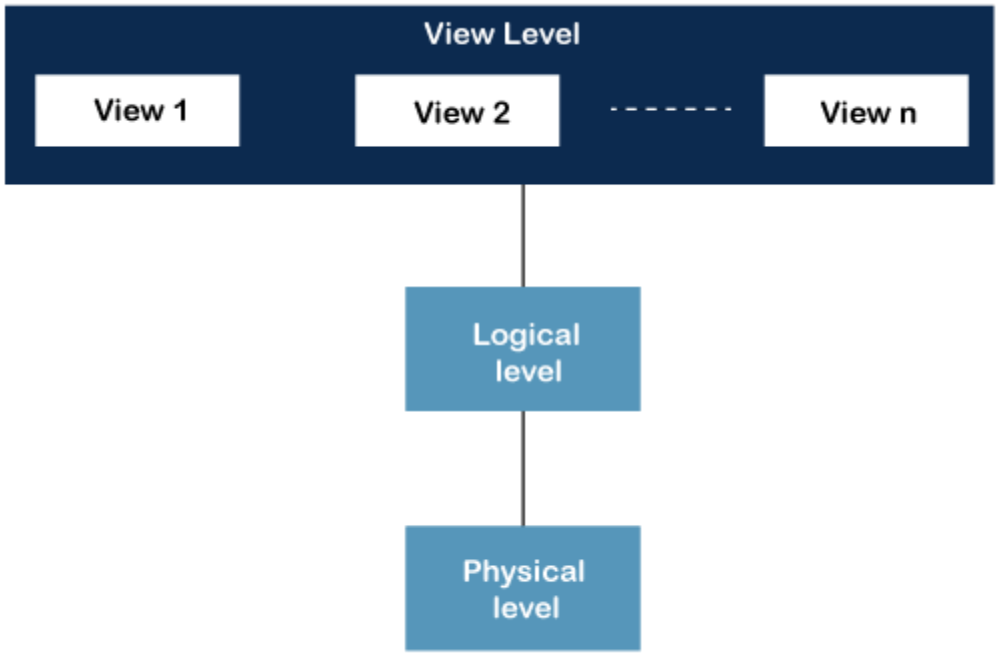
- Abstract class
- Abstract method

---

36) What are the levels of data abstraction?

There are **three** levels of data abstraction:

- **Physical Level:** It is the lowest level of data abstraction. It shows how the data is actually stored in memory.
- **Logical Level:** It includes the information that is actually stored in the database in the form of tables. It also stores the relationship among the data entities in relatively simple structures. At this level, the information available to the user at the view level is unknown.
- **View Level:** It is the highest level of data abstraction. The actual database is visible to the user. It exists to ease the availability of the database by an individual user.

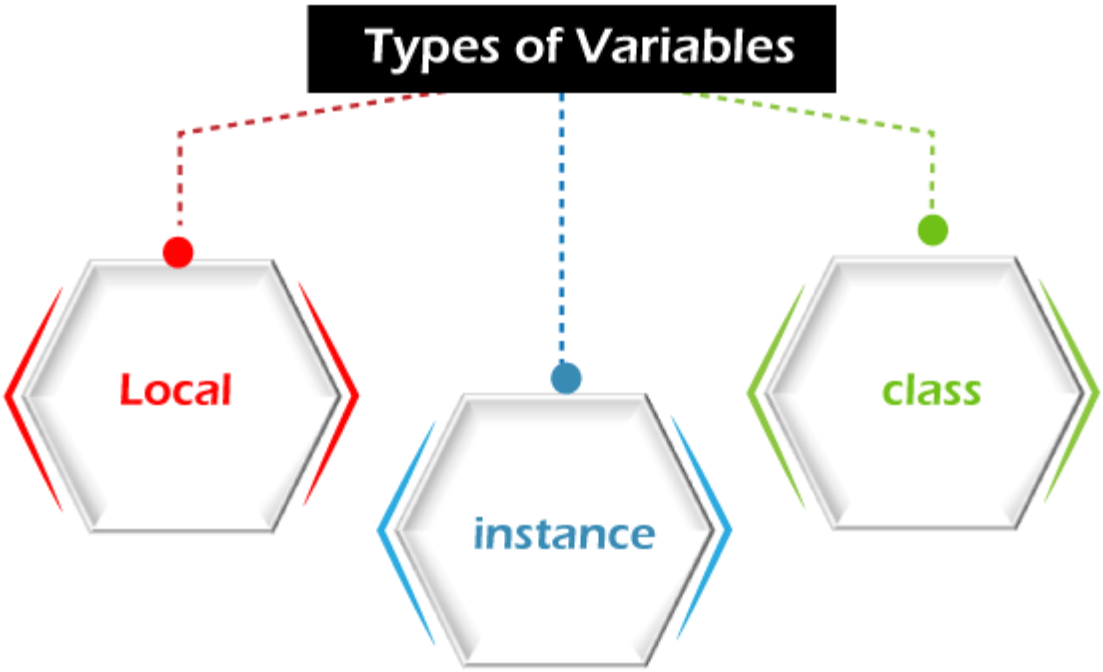


Levels of data abstraction

---

37) What are the types of variables in OOP?

There are three types of variables:



Instance Variable: It is an object-level variable. It should be declared inside a class but must be outside a method, block, and constructor. It is created when an object is created by using the new keyword. It can be accessed directly by calling the variable name inside the class.

Static Variable: It is a class-level variable. It is declared with keyword **static** inside a class but must be outside of the method, block, and constructor. It stores in static memory. Its visibility is the same as the instance variable. The default value of a static variable is the same as the instance variable. It can be accessed by calling the **class_name.variable_name**.

Local Variable: It is a method-level variable. It can be declared in method, constructor, or block. Note that the use of an access modifier is not allowed with local variables. It is visible only to the method, block, and constructor in which it is declared. Internally, it is implemented at the stack level. It must be declared and initialized before use.

Another type of variable is used in object-oriented programming is the **reference** variable.

Reference Variable: It is a variable that points to an object of the class. It points to the location of the object that is stored in the memory.

---

38) Is it possible to overload a constructor?

Yes, the constructors can be overloaded by changing the number of arguments accepted by the constructor or by changing the data type of the parameters. For example:

```
1. public class Demo
2. {
3. Demo()
4. {
5. //logic
6. }
7. Demo(String str) //overloaded constructor
8. {
9. //logic
10. }
11. Demo(double d) //overloaded constructor
12. {
13. //logic
14. }
15. //statements
16. }
```

---

39) Can we overload the main() method in Java also give an example?

Yes, we can also overload the main() method in Java. Any number of main() methods can be defined in the class, but the method signature must be different. Consider the following code.

```
1. class OverloadMain
2. {
3. public static void main(int a)  //overloaded main method
4. {
5. System.out.println(a);
6. }
7. public static void main(String args[])
8. {
9. System.out.println("main method invoked");
10. main(6);
```

```
}
}
```

---

40) Consider the following scenario:

If a class Demo has a static block and a main() method. A print statement is presented in both. The question is which one will first execute, static block or the main() method, and why?

JVM first executes the static block on a priority basis. It means JVM first goes to static block even before it looks for the main() method in the program. After that main() method will be executed.

```
class Demo
{
static                  //static block
{
System.out.println("Static block");
}
public static void main(String args[])  //static method
{
System.out.println("Static method");
}
}
```
