Define object-oriented programming.

Object-Oriented Programming (OOP) is a programming paradigm that places more emphasis on data and the concept of objects than on functions and logic when designing software. A software program is organized using OOP into straightforward, reusable classes of code blueprints that are used to produce distinct instances of objects or collection of objects.

Advantages of OOPs	Disadvantages of OOPs
OOPs provides enhanced code reusability.	The programmer should be well-skilled and should have excellent thinking in terms of objects as everything is treated as an object in OOPs.
The code is easier to maintain and update.	Proper planning is required because OOPs is a little bit tricky.
It provides better data security by restricting data access and avoiding unnecessary exposure.	OOPs concept is not suitable for all kinds of problems.
Fast to implement and easy to redesign resulting in minimizing the complexity of an overall program.	The length of the programs is much larger in comparison to the procedural approach.

Explain the main pillar or features of object-oriented programming language.

The four main features or pillars or principles of OOP are:

Encapsulation: The process of binding data and functions that operate on them so that no other part of the code can access this data except that function is known as encapsulation.

Abstraction: The process of hiding certain details and showing only essential information to the user is known as abstraction.

Polymorphism: The term "polymorphism" is the core concept that describes an object's, function's, or variable's capacity to assume various forms. It is a programming language's capacity to provide a single interface for a variety of diverse underlying data types and for various objects to react differently to the same message.

Inheritance: In object-oriented programming (OOP), inheritance enables programmers to independently modify original software via public classes and interfaces, specify a new implementation while preserving the same behaviours, and construct classes that are built upon existing classes.

Tell me some benefits of object-oriented programming language.

The advantages of OOP or object-oriented programming language are:

Modularity: Encapsulation makes it possible for things to be self-contained, which facilitates collaborative development and simplifies troubleshooting.

Code organization: OOP makes it easier to write modular code, which allows for easier code maintenance and improved program organization.

Scalability: OOP provides a clear code structure that is easy to understand because classes become grouped together in a hierarchical tree structure.

Security: Encapsulation allows for better data protection and security.

Explain structural programming.

Structured programming is a programming paradigm that makes it easier to write programs with clear code and reusable parts. It is a form of imperative programming in which nested loops, conditionals, and subroutines create control flow. It is based on a top-down approach.

The key features of structural programming are:

Sequence: Code lines are executed in the order they occur in the program, according to the sequence.

Repetition: Loops are employed to repeatedly run a block of code until a specific condition is met.

Selection: Depending on whether a condition is true or false, conditional statements are used to run distinct blocks of code.

Object-Oriented Programming	Structural Programming
Programming that is object-oriented is built on objects having a state and behavior.	A program's logical structure is provided by structural programming, which divides programs into their corresponding functions.
It follows a bottom-to-top approach.	It follows a Top-to-Down approach.
Restricts the open flow of data to authorized parts only providing better data security.	No restriction to the flow of data. Anyone can access the data.
Enhanced code reusability due to the concepts of polymorphism and inheritance.	Code reusability is achieved by using functions and loops.
Methods work dynamically, making calls based on object behavior and the need of the code at runtime.	Functions are called sequentially, and code lines are processed step by step.
Modifying and updating the code is easier.	Modifying the code is difficult as compared to OOPs.
Data is given more importance in OOPs.	Code is given more importance.

Describe a class in OOP.

A class is a blueprint or logical entity for creating objects in object-oriented programming (OOP), which provides initial values for state (member variables or attributes) and behaviour (member functions or methods). A class declaration essentially outlines the framework of what should be present and depicts how an object will look. Syntax:

```
class                                                                    {
field;
method;
}
```

How much memory does a class occupy?

Classes do not use memory. They merely serve as a template from which items are made. Now, objects actually initialize the class members and methods when they are created, using memory in the process.

What's an object in OOPs?

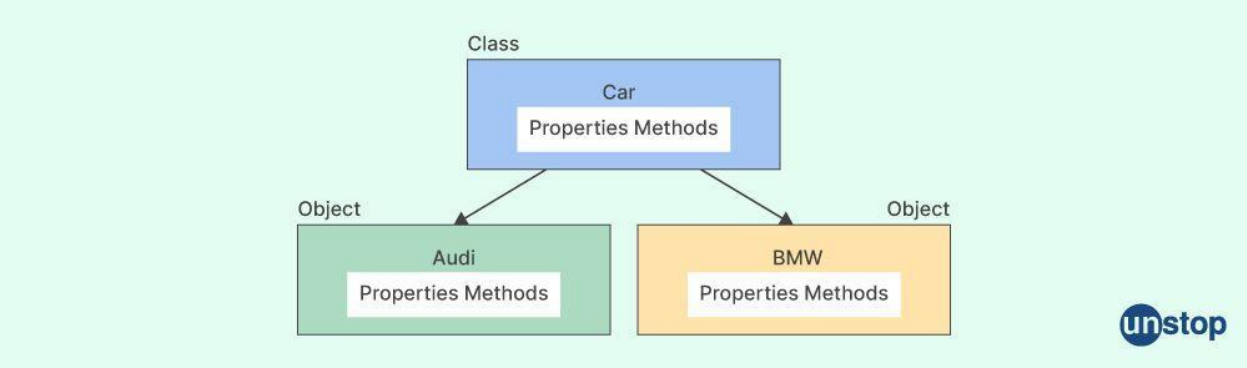
An object is a fundamental building piece in object-oriented programming (OOP). It is a standalone element made up of functions and attributes that make a specific kind of data useful. We can think of an object as a real-world entity with a state and behaviour. It is a member of a class, which serves as a model or prototype and lists the variables and operations that apply to all objects of a particular type. An object occupies some memory space, has an address and is a physical entity.

ClassName ReferenceVariable = new ClassName();

What are the differences between class and objects?

The differences between class and objects are:

CLASS	**OBJECT**
Class is a blueprint of objects.	A class' instances are objects.
It is a logical entity.	It is a physical entity.
A class is used to create multiple objects.	A particular object belongs to only one class.



Should the objects mandatorily be always created from the class?

No, it is not always necessary to create objects from a class in OOP. If the base class includes non-static methods, an object must be constructed. But no objects need to be generated if the class includes static methods. In this instance, you can use the class name to directly call those static methods.

What is the difference between a structure and a class in C++?

The structure is also a user-defined datatype in C++ similar to the class with the following differences:

- The major difference between a structure and a class is that in a structure, the members are set to public by default while in a class, members are private by default.
- The other difference is that we use **struct** for declaring structure and **class** for declaring a class in C++.

Tell me about a constructor in OOPs.

When an object is formed, a constructor is a special type of method that is automatically called in object-oriented programming. It is utilized to set up the newly generated object with some reliable values.

Setting default values for data members, allocating memory, and carrying out additional initialization procedures can all be done using constructors. The ability of constructors to accept parameters makes it possible to initialize attributes with relevant values. We can define constructors either within or outside the class definition. The different types of constructors are:

Parameterized Constructor

Copy Constructor

Static Constructor

Private Constructor

Dynamic Constructor

Conversion Constructor

Explain the meaning of destructor in C++.

In C++, destructors are class member functions that destroy an object. They are triggered whenever a class object exits its scope, such as at the conclusion of a function, the conclusion of a program, the calling of a delete variable, etc. Since destructors don't accept arguments and return nothing, they differ from typical member functions. Destructors are also named after their classes.

What do you mean by inheritance?

In OOP, inheritance is a strong idea that lets us implement subclasses that extend superclasses. The superclass' protected and public methods are all passed down to the subclass object due to inheritance. We can reuse code, link up various classes, and avoid any coding problems because of inheritance. The different types of inheritance are:

Single Inheritance

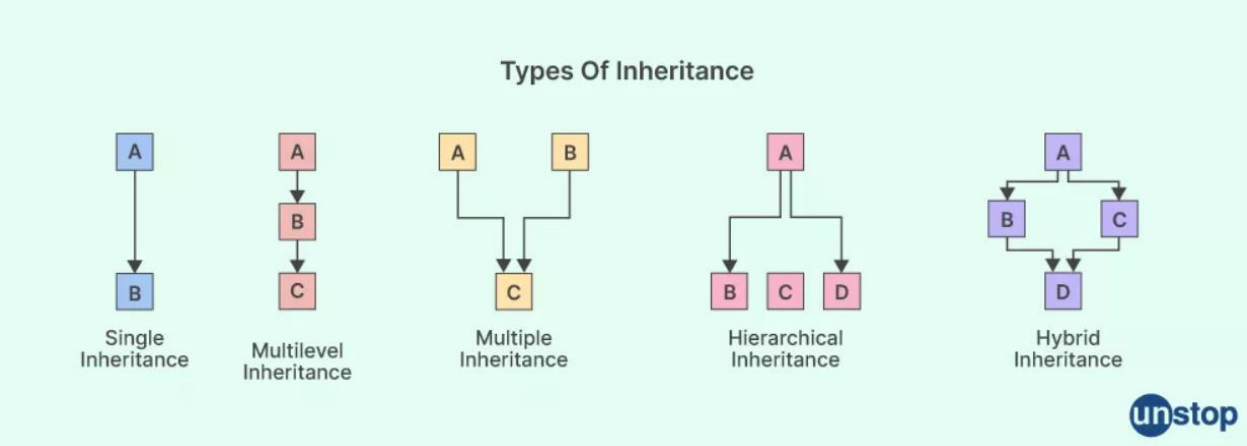
Multi-level Inheritance

Multiple Inheritance:

Hierarchical Inheritance

Hybrid Inheritance

Multipath inheritance



Tell me the differences between multiple and multi-level inheritances?

The major differences between multiple and multi-level inheritances are:

Multiple Inheritance:

A class inherits from multiple base classes.

It can make the system more complex.

Multiple inheritance has two class levels i.e., base class and derived class.

Multilevel Inheritance:

When a class inherits from multilevel inheritance, a derived class becomes the base class for a new class.

In many applications, it is well-liked and straightforward to use.

Multilevel inheritance has three class levels i.e., base class, intermediate class, and derived clas**s.**

Define hierarchical inheritance.

Hierarchical inheritance is a technique for transferring features from a parent class to a base, child, or subclass from an object-oriented perspective. The parent class or superclass is the class from which the attributes, i.e., the characteristics, are inherited. Hierarchical inheritance involves the descent of various classes from a single base or parent class.

Tell me about hybrid inheritance.

One of the most complex inheritance patterns accessible is hybrid inheritance - an inheritance type used in object-oriented programming that blends multiple inheritances. When employing multiple inheritances, a child class is descended from one or more combinations of single, hierarchical, and multilevel inheritances. This is known as a heterogeneous feature.

For programs to combine several types of inheritance, such as when combining a single inheritance with numerous inheritances or when multiple inheritances are combined inside a single program, hybrid inheritance is used.

Tell me some disadvantages of inheritance.

The limitations of inheritance in OOPs are:

- Improper use: Improper use of inheritance or less knowledge of inheritance may lead to wrong solutions. Sometimes, base class function data members are left idle, causing memory waste. Therefore inheritance requires careful implementation.
- Slower performance: Inherited functions may work slower than normal functions due to indirection.
- Inflexibility: Due to the possibility of changes to the base class method affecting all derived classes, inheritance might make code less adaptable. Due to this, it could be more challenging to add new functionality to the code.

Tell me some points of dissimilarities between structure and class.

Points of dissimilarities between structure and a class in OOPs are:

- A structure is a user-defined collection of data types or a collection of variables that combines logically related data items of different data types, whereas a class is a blueprint or a set of instructions to build a specific type of object.
- A class typically has type reference, but a structure typically has type value.
- A structure cannot be inherited, while a class can be inherited.

Define copy constructor.

A member function known as a copy constructor initializes an item using another object from the same class. To put it simply, a copy constructor is a constructor that produces an object by initializing it using a previously generated object of the same class. The copy constructor is used to copy the members of an existing object into a newly constructed object to initialize its members.

Define subclass in OOP.

A subclass in object-oriented programming is a class that is descended from the superclass or parent class. A subclass can utilize all the variables and methods of its superclass as though they had been declared within the subclass itself because it inherits all of them.

What do you mean by superclass?

The class from which numerous subclasses can be built is known as a superclass. The traits of a superclass are passed down to the subclasses. The parent class or base class function are other names for the superclass.

Tell me some limitations of object-oriented programming.

Object-oriented programming has certain limitations, such as:

- Large program size: Compared to programs developed using the procedure-oriented programming method, programs written using OOP may end up being larger in size.
- Complexity: The programs get more difficult as the amount of lines of code goes up and becomes a complex piece of code.

- CPU Intensive: Compared to other options, OOP is more CPU intensive.
- Flexibility: Some issues that arise need to be rectified with appropriate programming styles such as strategy or rational. In such circumstances, OOP is not the right option, and applying OOP will not be effective.

Explain garbage collection.

Garbage collection is the process of cleaning up resources that have been unused, or things that have outgrown their purpose, etc. The garbage collector recognizes when an object is no longer required and automatically destroys it, freeing up the memory space allotted to that object without harming objects that are still in use.

Explain finalize method.

Finalize method is used to perform cleanup activity before destroying any object. It is called by the garbage collector before destroying the objects from memory.

What do you know about the finally block?

Finally block is a block that is always executed, regardless of whether an exception is thrown or not and it follows the try and catch blocks in exception handling. The finally block is used to execute important code such as closing a connection or releasing resources.

The finally block is executed following the catch block if an exception is thrown and caught. The finally block is performed following the try block if an exception is thrown but not handled, which results in an unexpected program termination.

Features of the finally block are:

- In the exception handling chain, the finally block follows the try and catch blocks.
- The statements in the finally block will always be executed, regardless of whether there is an exception in the try block or not.
- Every essential statement which is contained in the finally block must be executed whether or not an exception occurs.
- Any other exception has the same behaviour as one that occurs in the finally block.

Tell me about try-catch block.

In Java, handling exceptions is done with a try-catch block. Inside the try block is the code or collection of statements that can throw an exception. The matching catch block takes care of an exception if it is raised.

SYNTAX:

```
try
{
// statement(s) that might cause exception
}


catch
{
// statement(s) that handle an exception
}
```

Explain the meaning of the exception.

An exception is a sudden occurrence that occurs while a program is running and prevents the regular execution of instructions. The program can detect it and deal with it. An object called the exception object is created when an exception occurs and it contains details about the exception.

Can you tell me if an exception is the same as an error in programming?

No, an exception is not the same as an error in programming.

An exception is a circumstance that takes place while the program is running and prevents it from running normally. Exceptions can be caught and dealt with within the code to make sure the application keeps functioning as intended.

An error is a serious condition that the program's code is unable to handle. They cannot be caught or handled. An error can never be recovered. It occurs at runtime and is always unchecked.

Tell me about the final variable.

The final keyword is used with classes, methods, and variables. An int, float, or other primitive data type variable cannot have its value modified when the final keyword is used with it.

Tell the points of dissimilarities between interfaces and abstract classes.

There are certain dissimilarities between interfaces and abstract class, such as:

Interfaces	Abstract class
An interface is a kind of code contract that must be implemented by a concrete class.	An abstract class is comparable to a regular class. It can also contain abstract methods, which are methods without a body.

An interface can only use final variables	Final, non-final, static, and non-static variables are all possible in an abstract class.
It can only have abstract methods and is a special type of class.	Abstract method and non-abstract method both can be present in an abstract class.
It also supports multiple inheritance.	It does not support multiple inheritance.

Explain data abstraction and encapsulation.

DATA ABSTRACTION

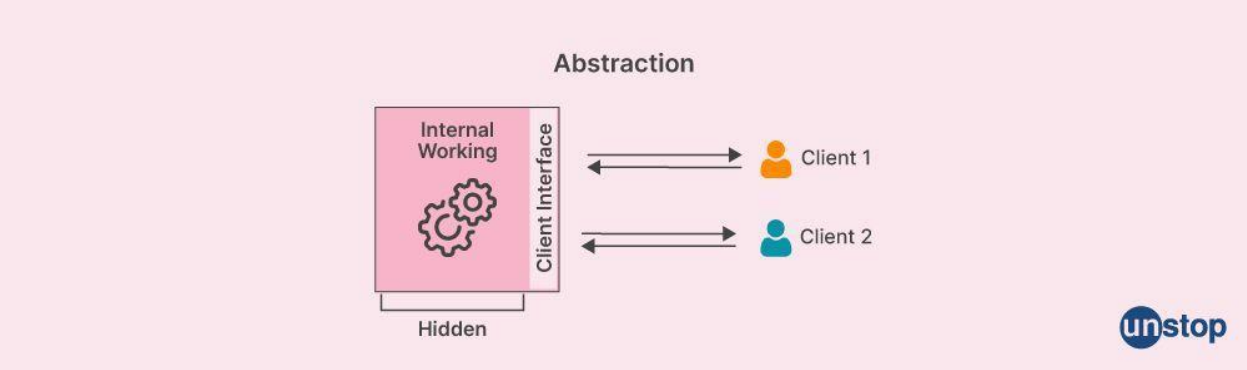
Abstraction is a process of extracting important information without involving the complete detail of the system or unnecessary details.

By presenting only the most necessary details, abstraction only displays relevant data.

It is focused mainly on what should be done by hiding the implementation details.

Abstraction hides complexity by giving us a more abstract picture.

It is achieved using abstract classes and interfaces.



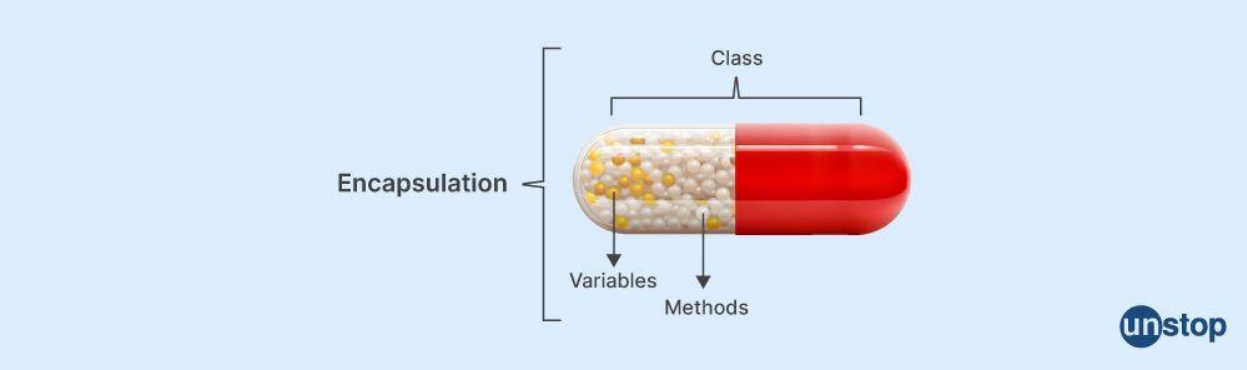
ENCAPSULATION

A technique for combining the code that manipulates the data with the data itself is called encapsulation.

A complex system can be made simpler for end users to use through encapsulation.

It wraps code and data for necessary information.

It hides internal work so that you can change it later.



Explain pure virtual function.

A pure virtual function is a virtual function or abstract function for which we do not have an implementation, we only declare it. It is employed to resemble the template, and the function is

carried out by the derived classes. Any class containing one or more pure virtual functions cannot be used to define any object.

Define virtual function.

A base class member function that is redefined in a derived class is referred to as a virtual function and it is used to instruct the compiler whether to conduct late binding or dynamic linking on the function.

Regardless of the type of reference (or pointer) used for the function call, virtual functions make sure that the right function is called for an object. The main purpose of them is to implement runtime polymorphism. The virtual keyword in the base class is used to declare virtual functions and runtime function call resolution is performed. In order to accomplish runtime polymorphism, virtual functions cannot be static and must be accessed using a pointer or reference of base class type.

Define access specifiers in C++.

A class member variable or function's level of access can be specified using access specifiers in C++. Here's a look at three different types of access specifiers:

Public: Everyone will have access to all class members stated with the public specifier.

Private: Only members of the class that have been explicitly designated as private can be accessed within the class.

Protected: All class members defined with the protected specifier can be accessed from within the class and any classes that it has derived.

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

Explain abstract class in C++.

A class in C++ is considered abstract if it has at least one pure virtual function. A class that is abstract serves as a suitable base class from which other classes can inherit. The definition of the pure virtual function must come from the classes that inherit the abstract class.

Some features of an abstract class are:

- The pure virtual function must be defined by the subclasses deriving from an abstract class; otherwise, the subclass would become an abstract class.
- Static methods, constructors, and abstract and non-abstract methods can be found in an abstract class.
- An abstract class can have final methods, which prevent the subclass from changing the method's body.

How will you achieve data abstraction?

We can achieve data abstraction in the following ways:

Abstraction using classes: A class can be used to group every data member and function into a solo unit with the help of access specifiers.

Abstraction in header files: Header files can be used to implement data abstraction in C++. For example, the pow() method present in math.h header file can be used to calculate the power of a number without knowing the underlying algorithm according to which the function is actually implemented.

Tell some advantages of data abstraction.

Some advantages of data abstraction are:

- It prevents duplication of software and increases code reusability.
- It simplifies design, optimization, and indexing, and ultimately increases the maintainability of the solution.
- It improves the privacy of an application or program as the user is only presented with relevant information and the unnecessary details are hidden.

- Data abstraction reduces data to its basic elements by further breaking it down to its most fundamental components, making it easier for users and programmers to understand the information.

Explain encapsulation.

Encapsulation is a core OOP concept that describes the grouping together of methods that operate on data. It protects user data, shields implementation information from other classes, and facilitates simpler reading, maintenance, and updating of code. Getter and setter methods, as well as access modifiers, are used to build encapsulation.

Discuss some advantages of encapsulation.

The advantages of encapsulation are:

- Hiding data: Encapsulation prevents users from accessing state values for all of the variables of a particular object, which provides data security.
- Flexibility: Programming is more adaptable due to encapsulation, which enables developers to set variables as read-only or write-only.
- Code reusability: Encapsulation makes it simple to modify and adapt code to meet new requirements.
- Increases code execution: Encapsulation results in an increase in the duration of the program execution, which can improve code execution

What is the difference between overloading and overriding?

A compile-time polymorphism feature called **overloading** allows an entity to have numerous implementations of the same name. Method overloading and operator overloading are two examples.

Overriding is a form of runtime polymorphism where an entity with the same name but a different implementation is executed. It is implemented with the help of virtual functions.

Can we overload the constructor in a class?

Yes We can overload the constructor in a class in Java. Constructor Overloading is done when we want constructor with different constructor with different parameter (Number and Type).

Can we overload the destructor in a class?

No, a destructor cannot be overloaded in a class. There can only be one destructor present in a class.

What are friend functions and friend classes?

Friend Functions: A friend function is a special function that is allowed to access private and protected data of a class, even though it's not a member of the class.

Friend Class: A friend class is a class that can access the private and protected members of another class. It's like allowing a trusted friend or a group of friends to see and change your personal information, which others cannot.

What is the virtual function and pure virtual function?

A virtual function is a function that is used to override a method of the parent class in the derived class. It is used to provide abstraction in a class.

- In C++ and C#, a virtual function is declared using the virtual keyword,
- In Java, every public, non-static, and non-final method is a virtual function.
- Python methods are always virtual.

Tell me about operator overloading.

OOP's operator overloading feature enables operators to be overloaded or redefined in order to deal with user-defined data types. It enables user-defined types to behave similarly to the fundamental primitive data types and provides class users with an easy user interface.

Define method overriding.

The ability to offer a specific implementation of a method that is already given by one of a subclass' superclasses or parent classes is known as method overriding in object-oriented programming.

Because the compiler doesn't actually know the type of object provided on compilation, it is also known as run-time polymorphism or dynamic binding.

Tell some differences between method overloading and method overriding.

Method Overloading:

Method overloading is the technique of having many methods in the same class with different parameters.

It is employed to make the program easier to read.

In case of method overloading, the parameter must be different.

Method Overriding:

When a subclass offers a particular implementation of a method that is already available in its parent class, it is known as method overriding.

It occurs in two classes that have an inheritance relationship.

The parameter must remain the same when a method is overridden.

Explain dynamic polymorphism.

The mechanism by which numerous methods with the same name and signature can be defined in the superclass and subclass is known as dynamic polymorphism. Runtime resolution occurs for the call to an overridden method. Thus, a call to an overridden method is resolved by a process known as dynamic polymorphism, commonly referred to as runtime polymorphism or dynamic method dispatch because it's executed at runtime rather than at compile-time.

It is accomplished by employing method overriding, in which a superclass reference variable is used to call an overridden method.

Features of dynamic polymorphism are:

- It takes place at runtime.
- It decides which method to execute at runtime.

- It is based on object orientation and is also known as dynamic binding.
- It enables separation between the interface and the implementation of a class hierarchy.
- It is achieved through virtual functions and function overriding.

What is static polymorphism?

When the decision of which technique to employ is determined during the compilation process, it is known as compile-time polymorphism or static polymorphism. In order to achieve this, operator or function overloading is used.

All the methods of static polymorphism get called or invoked during the compile time.

Features of static polymorphism are:

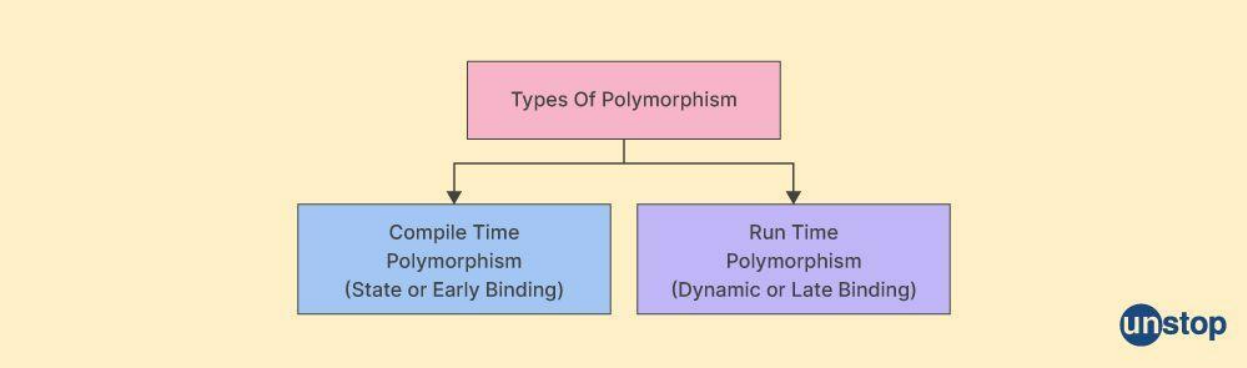
- It is resolved during compile time.
- It is also known as static binding polymorphism or compile-time polymorphism.
- Method and operator overloading is used to implement static polymorphism.
- All the methods of static polymorphism get called or invoked during the compile time.

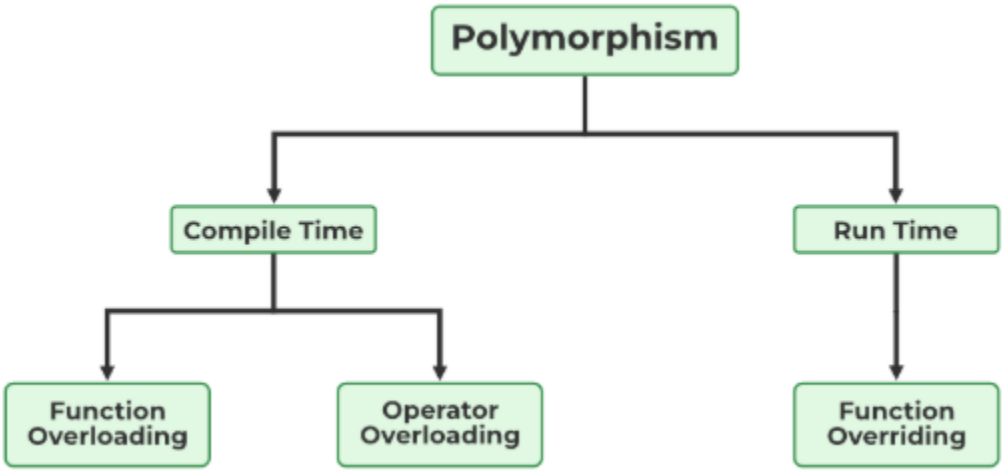
Define polymorphism and name the types of polymorphism.

Polymorphism in OOPs enables objects, functions, or variables to take multiple forms. Polymorphism is achieved through function overloading, function overriding, and virtual functions. In programming, it is generally used to implement inheritance.

The types of polymorphism in OOPs are:

- Compile-time polymorphism
- Run-time polymorphism





A) Compile-Time Polymorphism

Compile time polymorphism, also known as static polymorphism or early binding is the type of polymorphism where the binding of the call to its code is done at the compile time. Method overloading or operator overloading are examples of compile-time polymorphism.

B) Runtime Polymorphism

Also known as dynamic polymorphism or late binding, runtime polymorphism is the type of polymorphism where the actual implementation of the function is determined during the runtime or execution. Function overriding is an example of this method.

Tell me some advantages of polymorphism.

The advantages of polymorphism are:

- Reusable code: Once created, tested, and implemented, code and classes can be used again, saving programmers a tonne of work.
- Decreased coupling: Polymorphism aids in decoupling several functionalities from one another.
- Storage: Float, double, long, int, and other data types can all be stored in a single variable under a single variable name.
- Simple debugging: Simple debugging allows a coder to correct a program with fewer lines of code.
- Supports multiple data types: Polymorphism supports a single variable name for multiple data types.

Tell some features of the interface.

Some features of the interface are:

- Abstraction: An interface provides complete abstraction by using abstract methods. It specifies what a class can do, but not how it does it.
- Multiple inheritance: An interface helps in achieving multiple inheritance in languages that support it, by extending interfaces.
- Prototype of properties: An interface defines a prototype of properties of a family of classes, which ensures loose coupling.

Tell me the other name for the ternary operator.

The other name for a ternary operator is the conditional operator.

Draw the symbol of the scope resolution operator.

The scope resolution operator is denoted by a double colon (::).