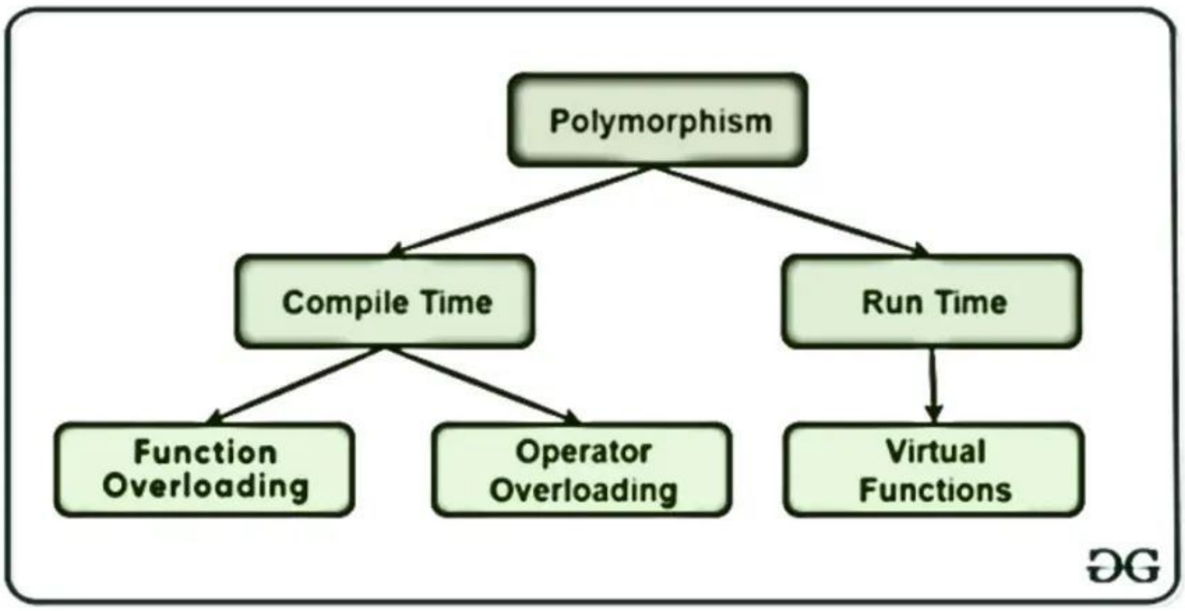


Types of Polymorphism:

1. Compile time polymorphism:
This type of polymorphism is achieved by function overloading or operator overloading.

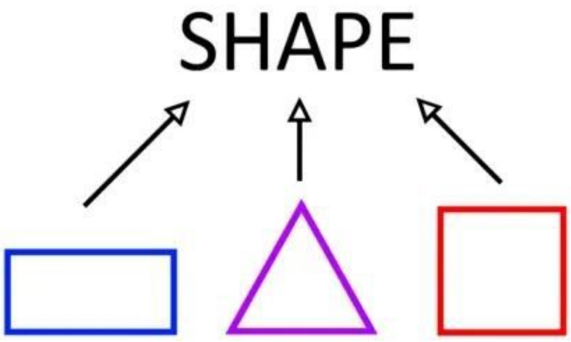
Function overloading:
When there are multiple functions with same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in number of arguments or/and change in type of arguments
multiple methods of same names performs different tasks within the same class.

2.Runtime polymorphism:
Runtime polymorphism refers to the process when a call to an overridden process is resolved at the run time.
This type of polymorphism is achieved by Function Overriding.

Function Overriding:
on the other hand, occurs when a derived class has a definition for one of the member functions of the base class.
methods having same name which can have different functionalities.
That base function is said to be overridden.

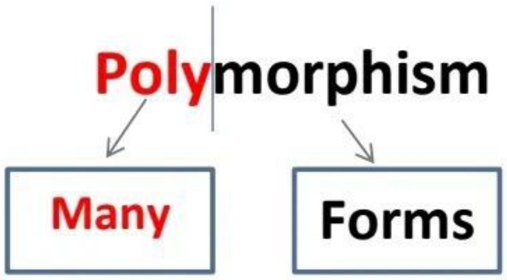
Polymorphism

Polymorphism is the ability of an object to take on **many forms.**
we can define polymorphism as the ability of a message to be displayed in more than one form.



A real-life example of polymorphism, a man at the same time is a father, a husband, an employee.

Another good real time example of polymorphism is water. Water is a liquid at normal temperature, but it can be changed to solid when it frozen, or same water changes to a gas when it is heated at its boiling point .Thus, same water exhibiting different roles is polymorphism.



polymorphism is mainly divided into two types:

- **Compile time Polymorphism (CTP)**

It is also called **static** polymorphism or **early binding.**

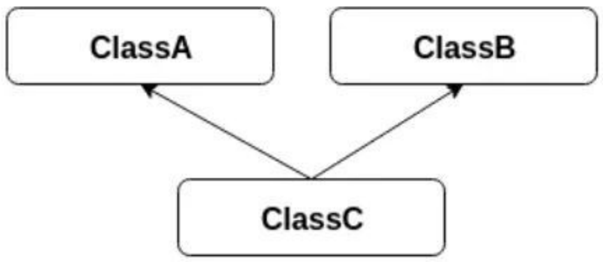
- **Runtime Polymorphism (RTP)**

It is also called **dynamic** polymorphism or **late binding.**

Multiple Inheritance

When a class inherits the properties and the behaviour of more than one class

Java, C#, most of high level language don't support Multiple Inheritance

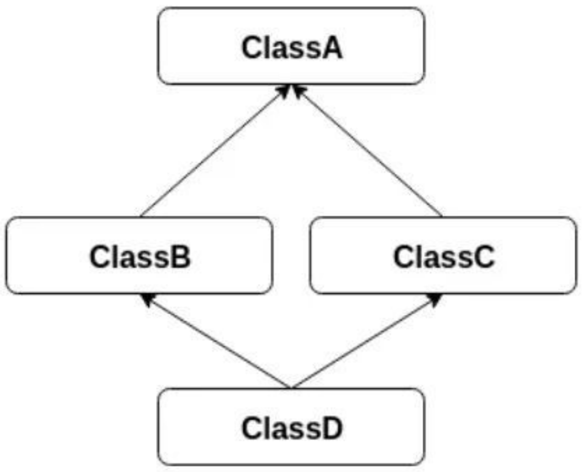


Multiple Inheritance

Hybrid Inheritance

Hybrid inheritance is a of inheritance is a combination of more than

one type of inheritance.



Hybrid Inheritance

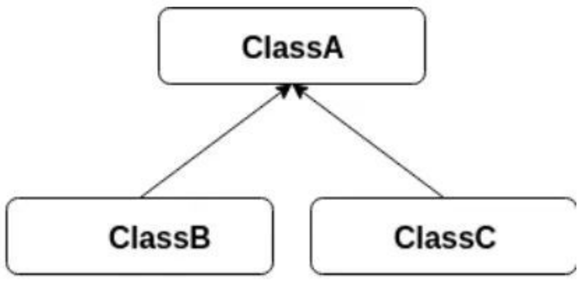
Why Java or C# don't support multiple inheritance?

because of following reasons –

Ambiguity Around The Diamond Problem Multiple inheritance does complicate the design and creates problem during casting, constructor chaining etc.

Hierarchical Inheritance

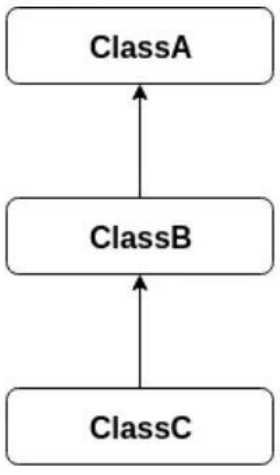
When more than class inherit properties and behaviour of only one class
In Hierarchical Inheritance there are only one parent and many child class



Hierarchical Inheritance

Multi-Level Inheritance

In this type of inheritance, a derived class is created from another derived class



Multilevel Inheritance

Inheritance

Inheritance is the procedure in which one class inherits the attributes and methods of another class.
In other words It is a mechanism of acquiring properties or behaviors of existing class to a new class

The Base Class, also known as the Parent Class is a class, from which other classes are derived.

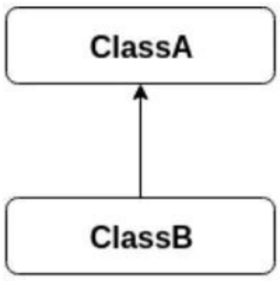
The Derived Class, also known as Child Class, is a class that is created from an existing class

There are four types of inheritance in OOP:

- Single Level Inheritance
- Hierarchical Inheritance
- Multi-Level Inheritance
- Multiple Inheritance
- Hybrid inheritance

Single Level Inheritance

When a class inherits properties and behaviour of only one class.
In other words, in single inheritance there is only one base class and only one sub class.



Single Inheritance

Abstraction

Allows to hide unnecessary data from the user. This reduces program complexity efforts.

it displays only the necessary information to the user and hides all the internal background details.

If we talk about data abstraction in programming language, the code implementation is hidden from the user and only the necessary functionality is shown or provided to the user.

In other words , it deals with the outside view of an object (Interface).

Eg.
-All are performing operations on the ATM machine like cash withdrawal etc. but we can't know internal details about ATM

-phone call we don't know the internal processing

We can achieve data abstraction by using
1. Abstract class
2. Interface

What is an abstract class?
Abstract class is that class which contains abstract method.

Abstract methods are those methods which have only declaration not the implementation.

An abstract class is declared with abstract keyword.

An abstract class can also contain non-abstract methods.

What are access specifiers ?

It allows us to restrict the scope or visibility of a package, class, constructor, methods, variables, or other data members.

There are three types of most **common access specifiers,** which are following.

- Private
- Public
- Protected

Public Modifiers :

means that class, variable or method is accessible throughout from within or outside the class, within or outside the package, etc.

It provides **highest level** of accessibility.

Private Modifiers :

means that class, variable or method is not accessible from within or outside the class, within or
outside the package, etc.

Private field or method can't be inherited to sub class.

This provides **lowest level** of accessibility.

Protected Modifiers :

means that class, variable or method is accessible from classes in the same package, sub-classes in
the same package, subclasses in other packages but not accessible from classes in other packages.

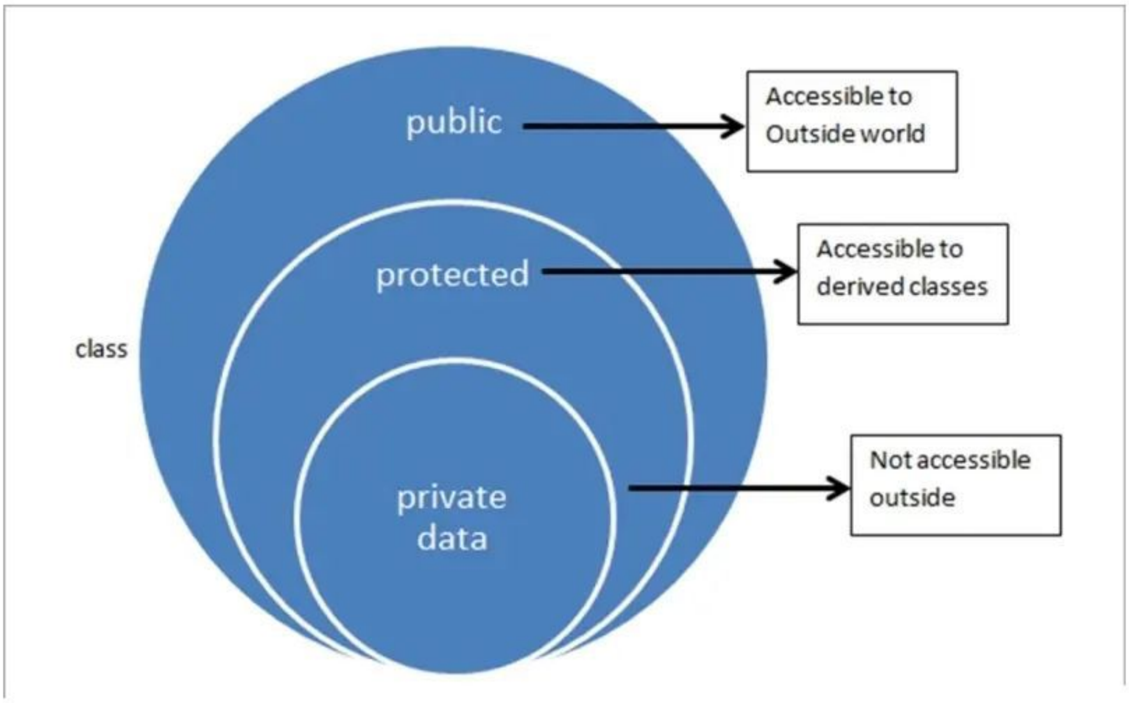
Access Modifiers	Accessible by classes in the same package	Accessible by classes in other packages	Accessible by subclasses in the same package	Accessible by subclasses in other packages
Private	NO	No	No	No
Public	Yes	Yes	Yes	Yes
Protected	Yes	NO	Yes	Yes

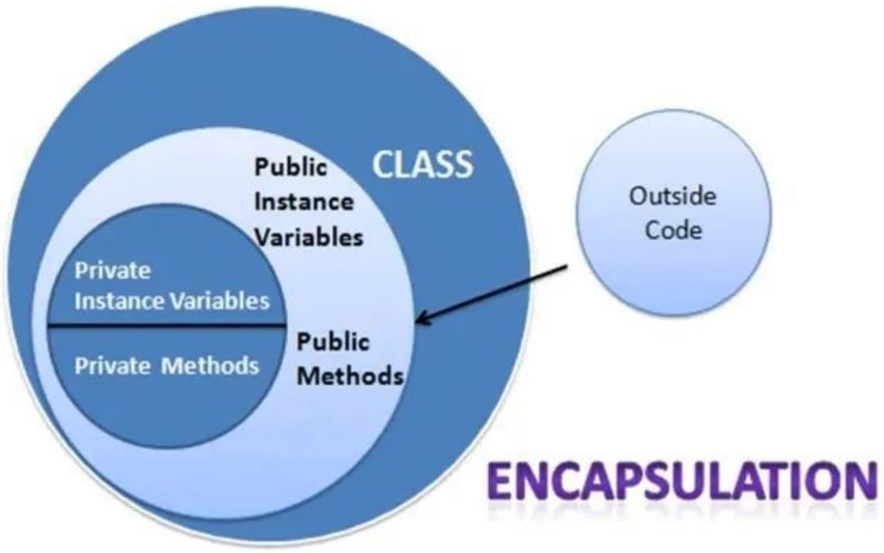
Encapsulation

The main advantage of encapsulation is that data is hidden and protected from randomly access by outside non-member methods of a class.

Encapsulation is the process of **binding** data and methods in a **single unit.**

In encapsulation, data(variables) are declared as private and methods are declared as public.

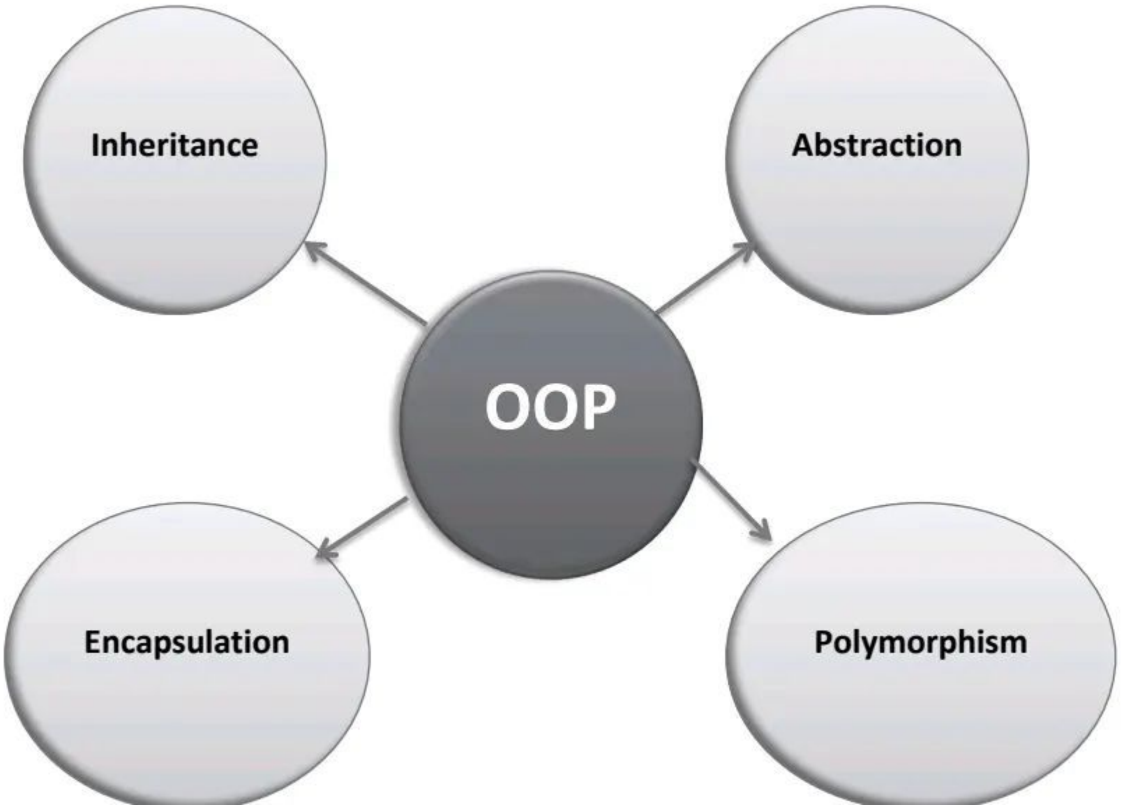




What is the difference between a class and a structure?

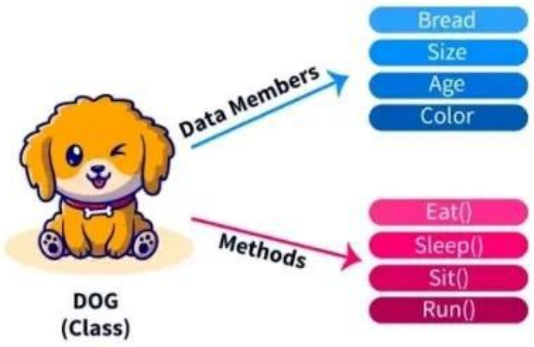
Class	Structure
1.Class is a collection of objects.	Structure is a collection of variables of different data types under a single unit
2. Class is used to combine data and methods together.	Structure is used to grouping data.
3. Class's objects are created on the heap memory.	Structure's objects are created on the stack memory.
4. A class can inherit another class.	A structure can't inherit another structure.
5. A class has all members private by default	A structure has all members public by default
6. Classes are ideal **for larger** or complex objects	Structures are ideal **for small** and isolated model objects

Following are the basic features of OOPs -



What is OOPs?

- It stands for Object-Oriented Programming.
- It is based on objects
- It follows Bottom-up programming approach.
- It is based on real world.
- It provides data hiding so it is very secure.
- It provides reusability feature.



What is a class?

A class is a collection of objects. Classes don't consume any space in the memory.

It is a user defined data type that act as a template for creating objects of the identical type.

A large number of objects can be created using the same class. Therefore, Class is considered as the blueprint for the object.

What is an object?

An object is a real world entity which have properties and functionalities. Object is also called an instance of class. Objects take some space in memory.

For eg .

Fruit is **class** and its **object** s are mango ,apple , banana
Furniture is **class** and its **objects** are table , chair , desk

What is the difference between a class and an object?

CREATE BY - ATUL KUMAR (LINKEDIN)

Class	Object
1. It is a collection of objects.	It is an instance of a class.
2. It doesn't take up space in memory.	It takes space in memory.
3. Class does not exist physically	Object exist physically.
4. Classes are declared just once	Objects can be declared as and when required