DATE
PAGE

Inheritance →

Inheritance is the capability of a class to derive properties and characterstics from another class is called Inheritance.

For example –
A child inherits the traits of his/her parents.

Purpose of Inheritance –

Inheritance promotes code reusability. When a derived class ~~can access~~ inherits the base class, then the derived class can access all the functionality, and the base class's code can be reused in the derived class.
It also allows programmers to create classes that are built upon existing classes.

C++ Inheritance →

The capability of a class to derive properties and characteristics from another class is called Inheritance.

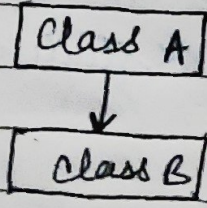
Includes :-

Base Class / Parent Class → The class whose properties are inherited by sub class / child class.

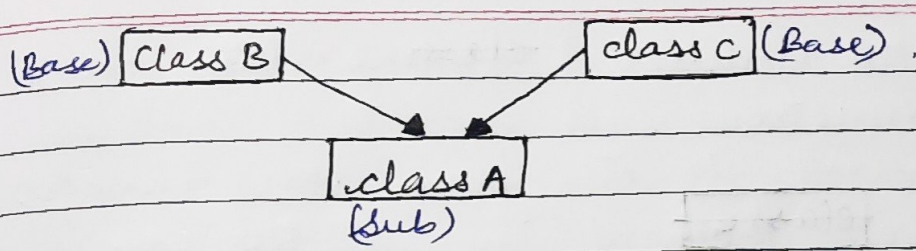
Sub Class / child class → The class that inherits properties from another class or Base / parent class.

Types Of Inheritance →

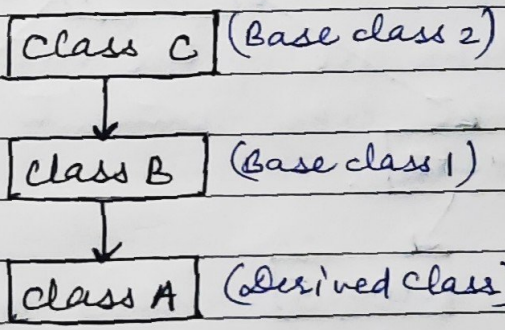
1) Single Inheritance —
In this, a class is allowed to inherit from only one class.
One sub class is inherited by only one Base class.



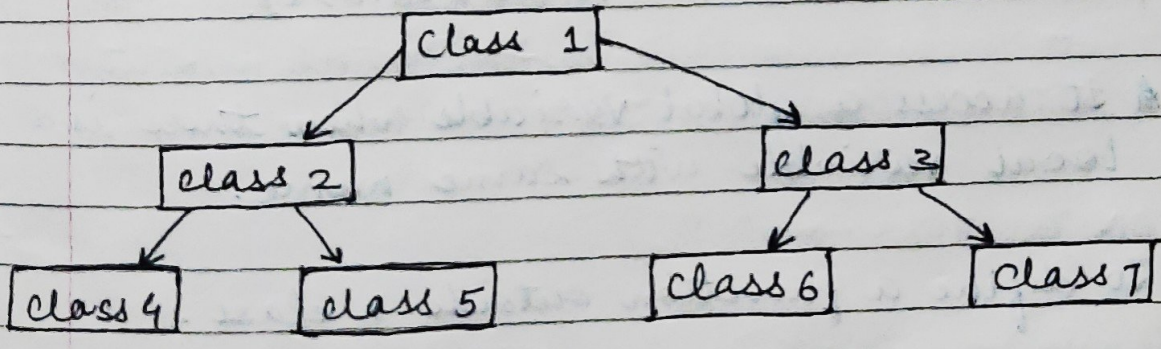
2) Multiple Inheritance —
In this, a class can inherit from more than one class.
One class is derived from more than one Base class



3) Multilevel Inheritance —
In this, a sub class / derived class is created from another derived class.

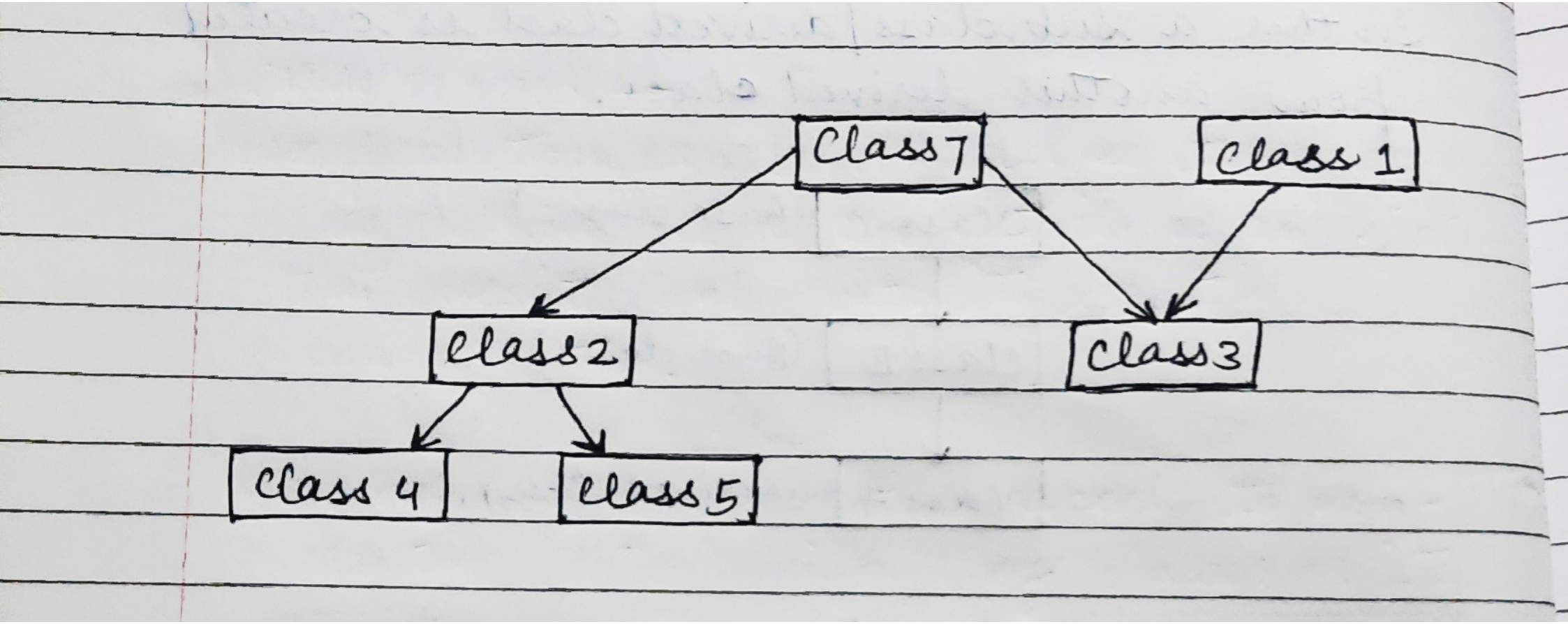


4) Hierarchical Inheritance —
In this, more than one sub / derived class is inherited from a single base class.



5) Hybrid Inheritance —
In this, more than one type of inheritance is combined.

For eg:- Hierarchical & Multiple Inheritance.


Class 7
class 1
class 2
class 3
class 4
class 5

★ "is a" Relationship →

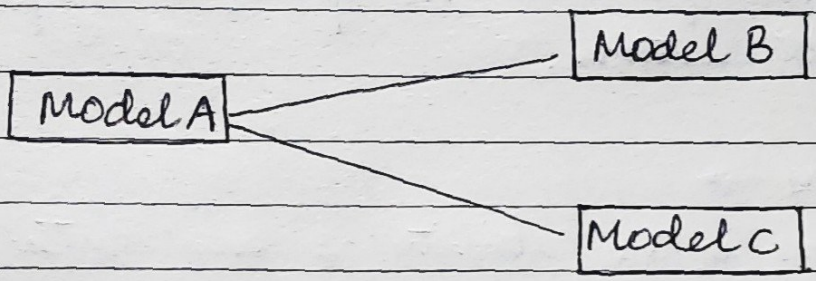
Inheritance is "IS-A" type of relationship. It is totally based on inheritance, which can be of two types Class Inheritance or Interface Inheritance.

Examples:-

- HOD is a staff member of college.
- Apple is a ~~fruit~~ fruit.
- Ferrari is a car.

★ Association and Aggregation →

Association – Association is defined as an organization of people with a common purpose.
It is a relationship between objects that describes an activity.
Eg:- A doctor can be associated with multiple patients



Aggregation – It is a collection or gathering of the things together. In other words, aggregation is a group, body or mass composed of many distinct parts or individuals.
Eg:- Phone number list is an example of aggregation

★ Interface and Abstract classes →

→ An interface describes the behaviour or capabilities of a class without applying to a particular implementation of that class.

This interface is implemented using abstract classes

→ Abstract classes are the classes whose objects cannot be created but they can be used as the base class for another classes.

A class is made abstract by declaring at least one of its function as a pure virtual function.

A pure virtual function is specified by placing "=0" in its declaration.

★ POLYMORPHISM →

Polymorphism is the ability of any data to be processed in more than one form.

Most common use of polymorphism in OOP is when:

→ A parent class reference is used to refer to a child class object.

It is the ability of an object or reference to take many forms in different instances.

Polymorphism implements the concept of

- Function Overloading
- Function over-riding
- Virtual functions.

- Polymorphism is a property through which any message can be sent to objects of multiple classes & every object can respond in a way, depending on the class properties.

★ Method Overloading & Method Over-riding.

Method Overloading	Method Over-riding
→ It is a compile-time poly-morphism.	→ It is a run-time poly-morphism.
→ It occurs within the class.	→ It is performed in 2 classes with inheritance relationship.
→ It helps increase the read-ability of the program.	→ Used to give specific implem-entation of method which is already provided by its parent class.
→ May or may not require inheritance.	→ Always needs inheritance.
→ In this, methods must have same name but different signatures.	→ In this, methods must have same name and same signatures.
→ Static binding is used.	→ Dynamic binding is used.
→ Poor performance due to compile-time polymorphism.	→ Better performance since binding of overridden methods is done at runtime.

★ Compile Time & Run Time Polymorphism →

Basis of comparison	Compile Time	Run Time
Basic	Compile time polymo-rphism means binding is occuring at compile time.	Runtime polymorphism where at run time we came to know which method is going to invoke.
Static/Dynamic Binding	It can be achieved through static binding.	It can be achieved through dynamic binding.
Inheritance	Inheritance is not involved.	Inheritance is involved.
Example	Method Overloading	Method overriding.

★ Virtual functions →

Virtual function is a member function which is declared within a ~~class~~ base class and is overridden by a derived class.

- Virtual functions ensure that the correct function is called for an object.
- They are mainly used to ~~achieve~~ achieve Runtime Polymorphism.

- functions are declared with a "virtual" keyword in the base class.
- They cannot be static.
- A virtual function can be a friend function of another class.
- Accessed using pointers.
- Virtual functions are always ~~derived~~ defined in the base class & overridden in a derived class.

★ Friend functions →
friend function can access private & protected members.
It can be:
1) A member of another class.
2) A global function.

- A friend function is a special function and is a non-member function of a class.
- By declaring a function as a friend, all the access permissions are given to the function.
- It can be declared in any section of the class i.e. public, private or protected.

★ Static function →
When a function inside a class is declared as static it can be accessed outside the class using the class name & scope resolution (::) operator, without creating objects.

- Static function has access to only the static members of the class, we cannot call the non-static functions inside it.

★ Friend Class →

Friend class can access private and protected members of the other class in which it is declared as a friend.

- It sometimes allows, private members of any other class to be accessed by ~~the~~ a particular class.

Exception Handling →

Exceptions are runtime anomalies or abnormal conditions that a program encounters during its execution.

• Why exception handling? →

1) Seperation of Error handling code from normal code →

In normal code, there are always if-else conditions to handle errors. These codes to handle errors, get mixed up with the normal flow, which make the code less readable & maintainable.

With try/catch blocks, the code for error handling becomes seperate from normal flow.

2) Grouping of Error types →

We can create a hierarchy of exception objects, group exceptions in namespaces or classes & categorize them according to their types. This helps find error / exceptions faster.

★ Multi-threading in C++ →

It is a specialized form of multitasking, and multitasking is a feature that allows your computer to run two or more programs concurrent

There are 2 types of multitasking:-

i) process-based
ii) thread-based.

Process based handles the current execution of the programs.

Thread based deals with the current execution of the ~~same~~ pieces of the same program.