Friend Class

A Friend Class can access private and protected members of other class in which it is declared as Friend.

Example:-

```
1.  class A {
2.  private:
3.      int a;
4.
5.  public:
6.      A() { a = 0; }
7.      friend class B; // Friend Class
8.  };
9.
10.     class B {
11. private:
12.         int b;
13.
14.     public:
15.         void showA(A& x)
16.         {
17.             // Since B is friend of A, it can access
18.             // private members of A
19.             std::cout << "A::a=" << x.a;
```

```
20.     }
21.   };
```

Pointers to Object

A Variable that holds an Address value is called a Pointer Variable or simply Pointer,Objects can also have an address, so there is also a pointer that can point to the address of an object.

Syntax:-
Class_name *object_pointer_name;

Ex:-

```
1. class Box {
2.   public:
3.      // Constructor definition
4.      Box(double l = 2.0, double b = 2.0, double h = 2.0)
   {
5.         cout <<"Constructor called." << endl;
6.         length = l;
7.         breadth = b;
8.         height = h;
9.      }
10.        double Volume() {
11.           return length * breadth * height;
12.        }
13.
14.    private:
15.        double length;     // Length of a box
16.        double breadth;    // Breadth of a box
17.        double height;     // Height of a box
18.  };
19.
```

```
20.  int main(void) {
21.     Box Box1(3.3, 1.2, 1.5);     // Declare box1
22.     Box Box2(8.5, 6.0, 2.0);     // Declare box2
23.     Box *ptrBox;                 // Declare pointer to a
    class.
24.
25.     // Save the address of first object
26.     ptrBox = &Box1;
27.
28.     // Now try to access a member using member access
    operator
29.     cout << "Volume of Box1: " << ptrBox->Volume() <<
    endl;
30.
31.     // Save the address of second object
32.     ptrBox = &Box2;
33.
34.     // Now try to access a member using member access
    operator
35.     cout << "Volume of Box2: " << ptrBox->Volume() <<
    endl;
36.
37.     return 0;
38.  }
```

Pointers to members:-

1.Pointer to a variable of a class:-

```
1. class Data{
2.       public:
3.       int a,b;
4.       void print()
5.       {
```

```
6.              cout<<"a = "<<a<<endl;
7.        }
8. }
9.
10.int main()
11.{
12.       Data d,*dp;
13.       dp=&d;
14.       int data::*ptr=&Data::a;
15.
16.       d.*ptr=10;
17.       d.print();
18.
19.       dp->*ptr=20;
20.       dp->print()
21.}
```

Output:-
10
20

2.Pointer to a function:-
Syntax:-
return type(class_name::*ptr_name)(argument type)=&class_name::fun_name

this pointer

Every object in C++ has access to its own address through an important pointer called this pointer.
Friend function doesn't have 'this' pointer because friends are not members of class.

```
1. class Box{
2.      private:
3.          int l,b,h;
4.      public:
```

```
        void set(int l, int b, int h){
         this->l = l;
         this->b = b;
         this->h = h;
        }
};

int main(){
   Box b;
   b.set(5,6,2);
}
```

Inheritance

The capability of a class to derive properties and characteristics from another class is called **Inheritance**.
Sub Class: The class that inherits properties from another class is called Sub class or Derived Class.
Super Class: The class whose properties are inherited by sub class is called Base Class or Super class.

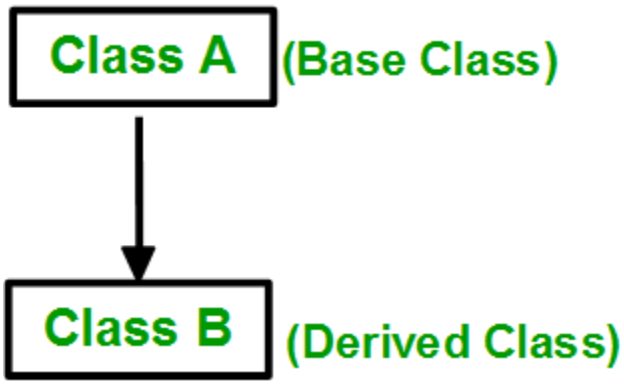
Syntax:

```
class subclass_name : access_mode base_class_name
{
```

```
3.  //body of subclass
4. };
```

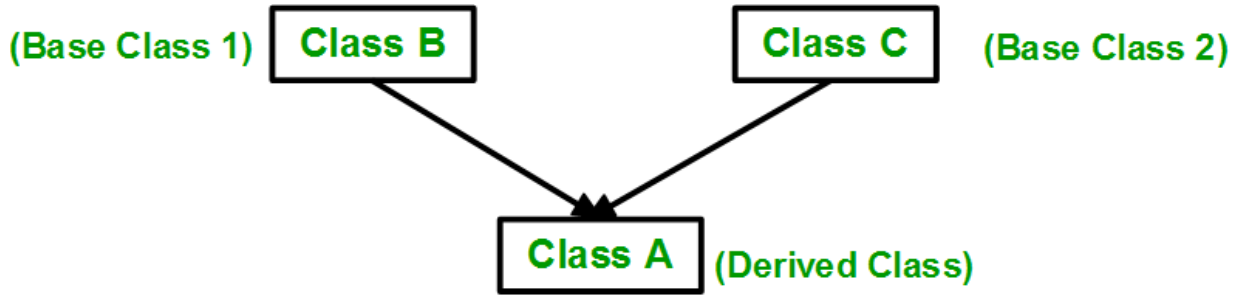
Single Inheritance:-

Single inheritance is one type of inheritance in which the derived class inherits only one base class.



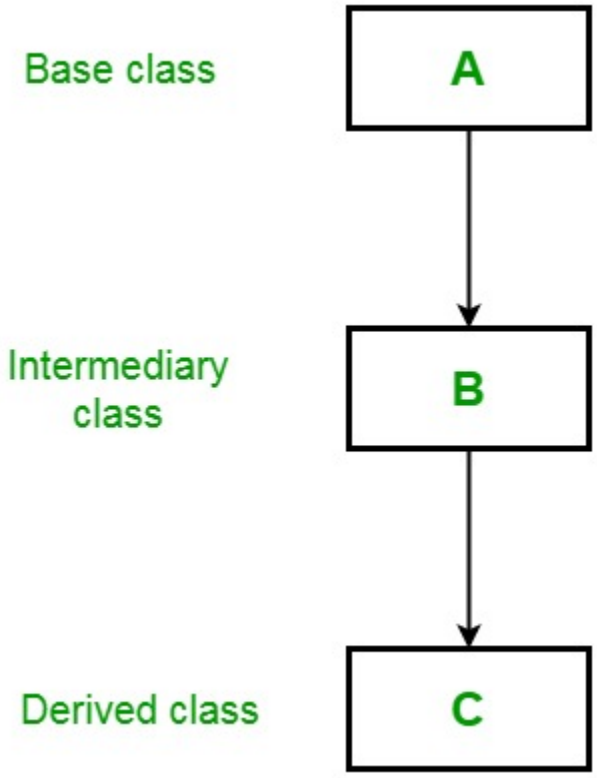
Multiple Inheritance:-

Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes



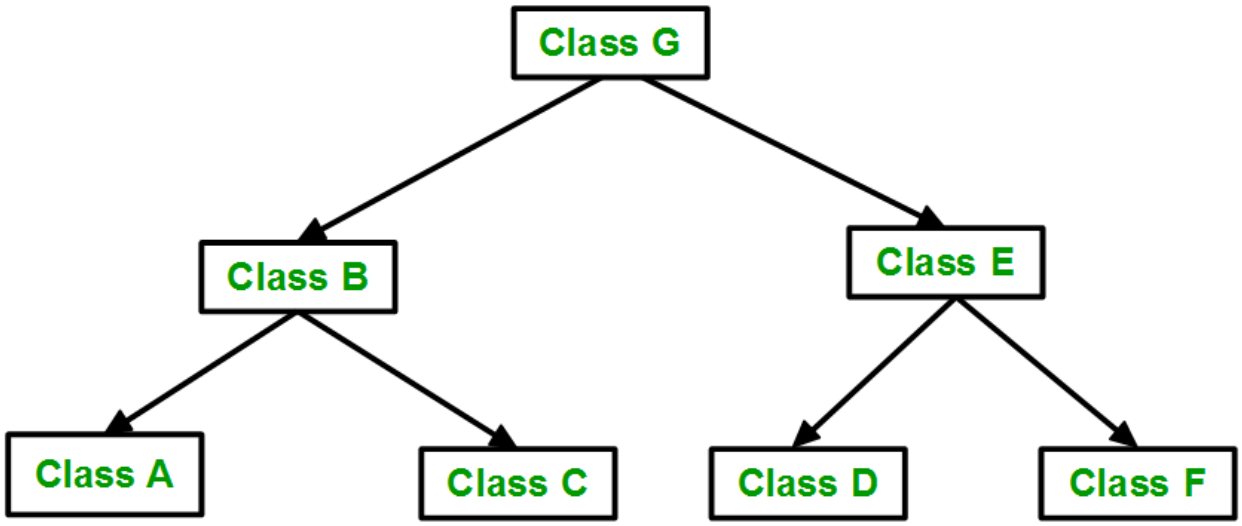
Multilevel Inheritance:-

Multilevel Inheritance is a property wherein an object of one class possesses the properties of another class and can further inherit the properties to other classes.



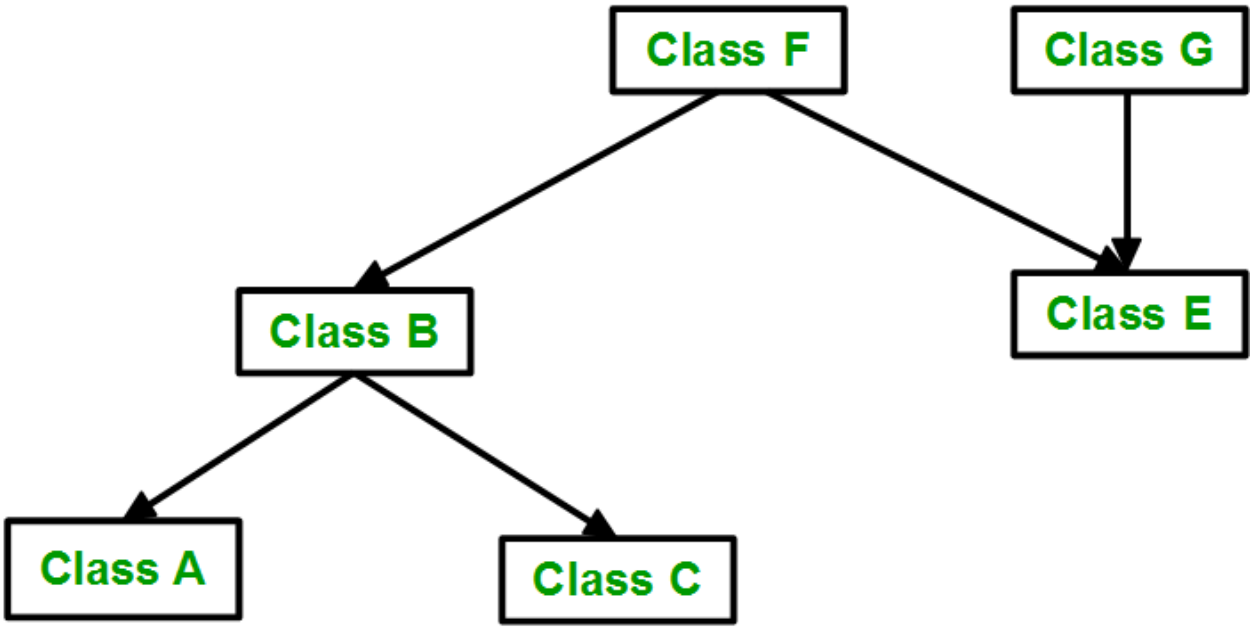
Hierarchical Inheritance:-

If more than one class is inherited from the base class, it's known as hierarchical inheritance.



Hybrid Inheritance:-

Hybrid inheritance in C++ is the inheritance where a class is derived from more than one form or combinations of any inheritance.



Constructor and Destructor in Inheritance:-

First child class constructor will run during creation of object of child class,but as soon as object is created child constructo run and it will call constructor of its parent class and after the execution of parent class constructor it will resume its constructor execution.

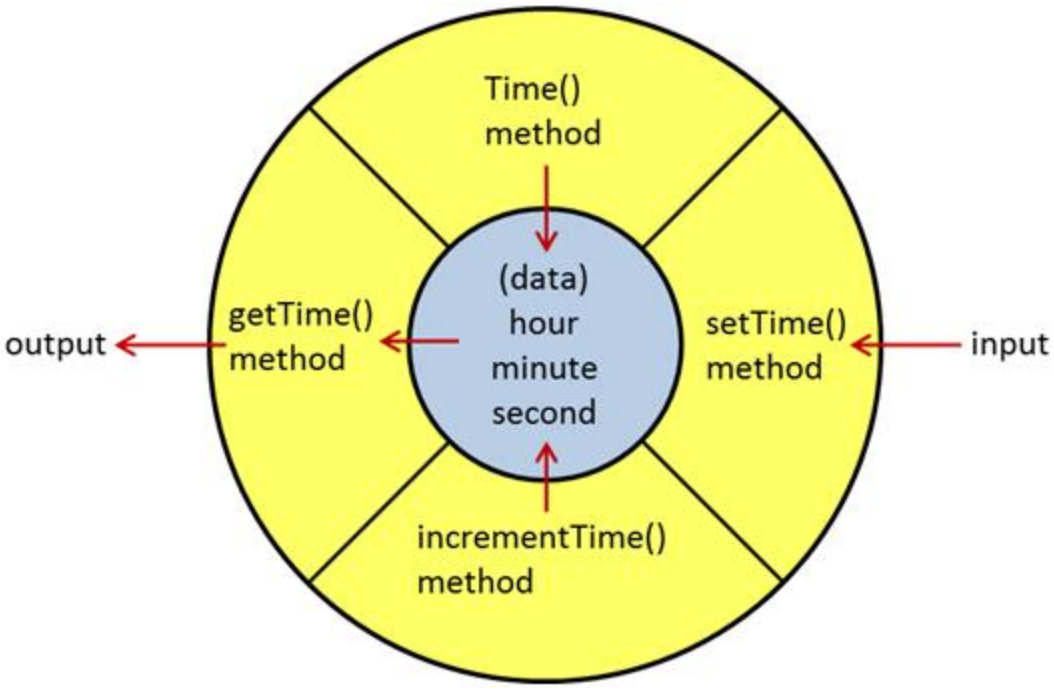
While in case of destructor, first child destructor executes then parent destructor get executed.

Data Hiding

It is a concept of oops which confirms the security of members of class from unauthorized access. It is a technique of protecting data members from being manipulated from any other source.
It can be achieved through encapsulation.

Encapsulation

Encapsulation is defined as wrapping up of data and information under a single unit. In Object Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulates them.



Ex:-

```
1. class MyAccount
2. {
3.      private:
4.      float balance;
5.      public:
6.      MyAccount(float amount)
7.      {
8.           balance=amount;
9.      }
10.     void withdraw(float amount)
11.     {
12.          if(amount>balance)
13.          {
14.               cout<<" Balance is less than
   Amount"<<endl;
15.          }
16.          else
```

```
17.          {
18.               balance-=amount;
19.          }
20.     }
21.     float get_balance()
22.     {
23.          cout<<"Balance: Rs."<<balance<<endl;
24.     }
25.     void Add_Money(float amount)
26.     {
27.          balance+=amount;
28.          cout<<"Current Balance : "<<balance<<endl;
29.     }
30. }
31.
```