INHERITANCE

Inheritance:- **The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important feature of Object Oriented Programming.**

We use two class

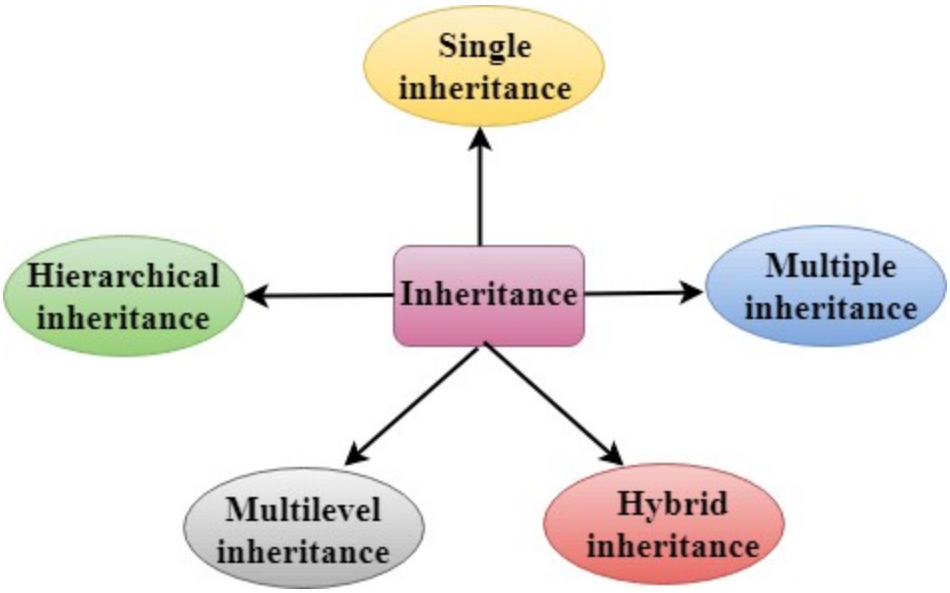
i. **Base class**
ii. **Derive class**

Base class:-The class whose properties are inherited by Derive class (sub class) is called Base Class or Super class or parent class.

Derive class:- The class that inherits properties from another class is called Derived Class or Sub class or child class.

Types Of Inheritance:-**In c++ inheritance basically five types.**

1. **Single inheritance**
2. **Multilevel inheritance**
3. **Multiple inheritance**
4. **Hierarchical inheritance**
5. **Hybrid inheritance**



Modes of Inheritance:-

1. **Public mode**: **If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.**

2. **Protected mode: If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class.**

3. **Private mode: If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class.**

Note:- The private members in the base class cannot be directly accessed in the derived class, while protected members can be directly accessed. For example, Classes B, C and D all contain the variables x, y and z in below example. It is just question of access.

1. **Single inheritance:-** When a single class is derived from a single parent class, it is called **Single inheritance**. It is the simplest of all inheritance.

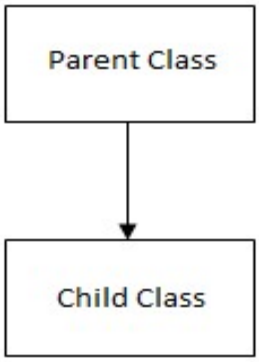


Fig: Single inheritance

Syntax:-

```
class base_classname
{
   properties;
   methods;
};

class derived_classname : visibility_mode base_classname
{
   properties;
   methods;
};
```

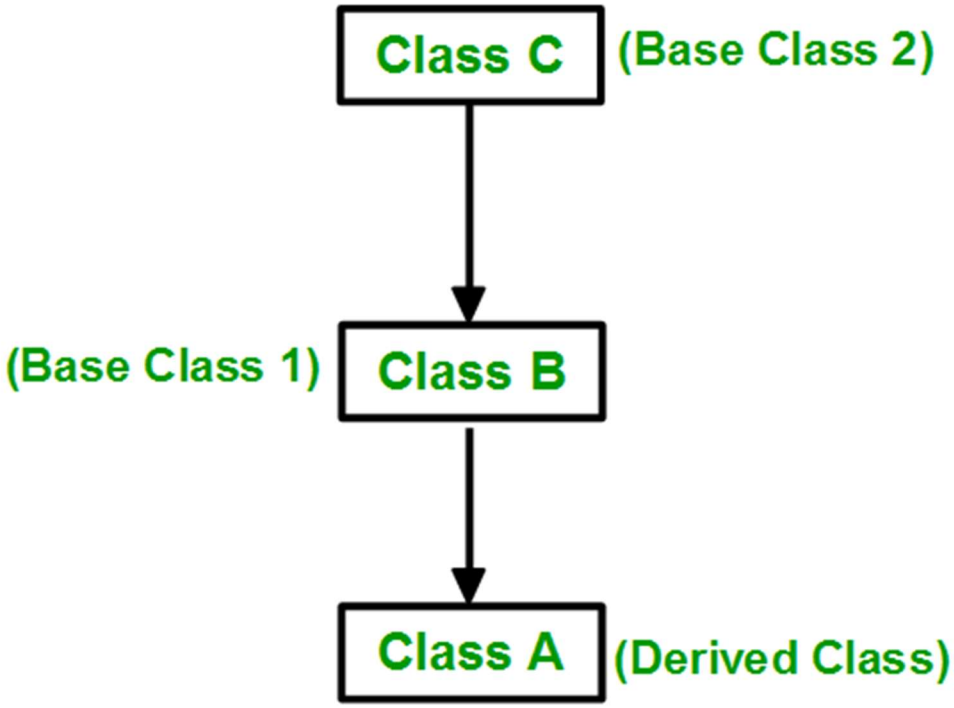
Example:-

```
#include <iostream>
using namespace std;

class Vehicle          // base class
```

```
 {
  public:
    Vehicle()
    {
      cout << "This is a Vehicle." << endl;
    }
};

// sub class derived from two base classes
class Car: public Vehicle
{

};

// main function
int main()
{
    // creating object of sub class will
    // invoke the constructor of base classes
    Car obj;
    return 0;
}
```

Output:- **This is a vehicle.**

2. ***Multilevel inheritance:-*** **In this inheritance, a derived class is created from another derived class.**



Syntax:-

```
class base_classname
{
   properties;
   methods;
};

class intermediate_classname:visibility_mode base_classname
{
   properties;
   methods;
};

class child_classname:visibility_mode intermediate_classname
```

```
{
    properties;
    methods;
};
```

Example:-

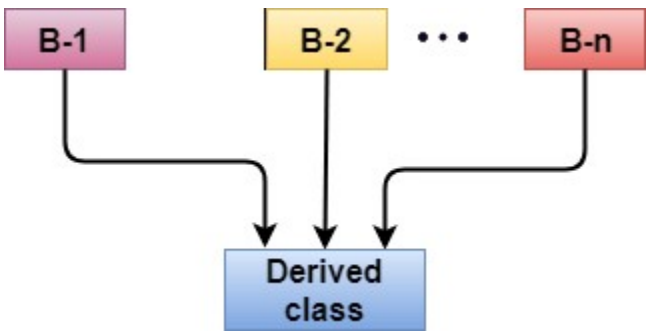
```
#include <iostream>
using namespace std;
class Vehicle              // base class
{
 public:
   Vehicle()
   {
    cout << "This is a Vehicle" << endl;
   }
};
class fourWheeler: public Vehicle
{  public:
   fourWheeler()
   {
    cout<<"Objects with 4 wheels are vehicles"<<endl;
   }
};
// sub class derived from two base classes
class Car: public fourWheeler{
```

```
  public:
    car()
    {
      cout<<"Car has 4 Wheels"<<endl;
    }
};
 int main()        // main function
{
   //creating object of sub class will
   //invoke the constructor of base classes
   Car obj;
   return 0;
}
```

output:

```
This is a Vehicle
Objects with 4 wheels are vehicles
Car has 4 Wheels
```

3. **Multiple inheritance:-** Multiple inheritance is the process of deriving a new class that inherits the attributes from two or more classes.



Syntax:-

```
class base_class1
{
   properties;
   methods;
};

class base_class2
{
   properties;
   methods;
};
... ... ...
... ... ...
class base_classN
{
   properties;
   methods;
};

class derived_classname : visibility_mode base_class1, visibility_mode
base_class2,... ,visibility_mode base_classN
{
   properties;
   methods;
};
```

Example:-

```
#include <iostream>
#include <conio.h>
using namespace std;
class liquid
{
   float specific_gravity;
   public:
      void input()
      {
         cout<<"Specific gravity: ";
         cin>>specific_gravity;
      }
      void output()
      {
         cout<<"Specific gravity: "<<specific_gravity<<endl;
      }
};

class fuel
{
   float rate;
   public:
      void input()
      {
         cout<<"Rate(per liter): $";
```

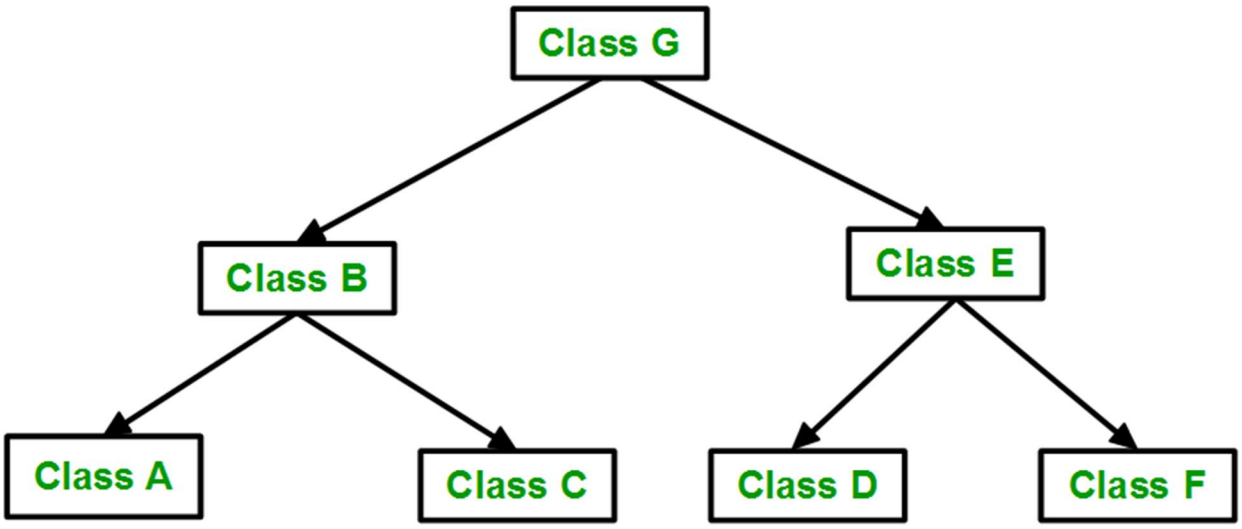
```
        cin>>rate;
    }
    void output()
    {
        cout<<"Rate(per liter): $"<<rate<<endl;
    }
};

class petrol: public liquid, public fuel
{
  public:
    void input()
    {
        liquid::input();
        fuel::input();
    }
    void output()
    {
        liquid::output();
        fuel::output();
    }
};

int main()
{
  petrol p;
```

```
    cout<<"Enter data"<<endl;
    p.input();
    cout<<endl<<"Displaying data"<<endl;
    p.output();
    getch();
    return 0;
}
```

Output:-

```
Enter data
Specific gravity: 0.7
Rate(per liter): $0.99

Displaying data
Specific gravity: 0.7
```

4. ***Hierarchical inheritance:-*** In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.



Syntex:-

```
class base_classname
{
   properties;
   methods;
};
class derived_class1:visibility_mode base_classname
{
   properties;
   methods;
};
class derived_class2:visibility_mode base_classname
{
   properties;
   methods;
};
... ... ...
... ... ...
class derived_classN:visibility_mode base_classname
{
   properties;
   methods;
};
```

Example:-

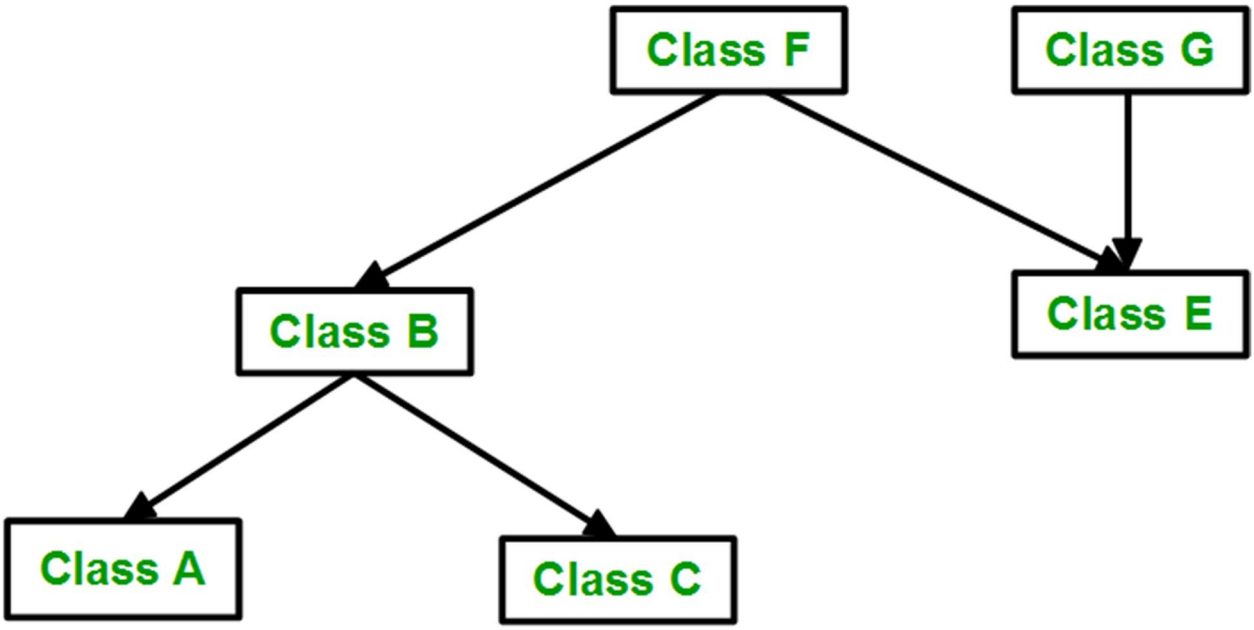
```
#include <iostream>
using namespace std;
class Vehicle       // base class
{
  public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};
class Car: public Vehicle            // first sub class
{

};
class Bus: public Vehicle                    // second sub class
{

};

// main function
int main()
{
   // creating object of sub class will
```

```
    // invoke the constructor of base class
    Car obj1;
    Bus obj2;
    return 0;
}
```

Output:

```
This is a Vehicle
This is a Vehicle
```

5. ***Hybrid inheritance***:- **Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.**
   Below image shows the combination of hierarchical and multiple inheritance:



Syntax:-

```
class A
```

```
{
    .....
};
class B : public A
{
    .....
} ;
class C
{
    .....
};
 class D : public B, public C
{
    .....
};
```

Example:-

```
#include <iostream>
using namespace std;
class Vehicle            // base class
{
  public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};
```

```
 class Fare            //base class
{
   public:
   Fare()
   {
      cout<<"Fare of Vehicle\n";
   }
};
 class Car: public Vehicle          // first sub class
{

};
 class Bus: public Vehicle, public Fare      // second sub class

{

};

int main()            // main function
{
   // creating object of sub class will
   // invoke the constructor of base class
```

```
    Bus obj2;
    return 0;
}
```

Output:-

This is a Vehicle

Fare of Vehicle

Created by:- Ajay kumar verma