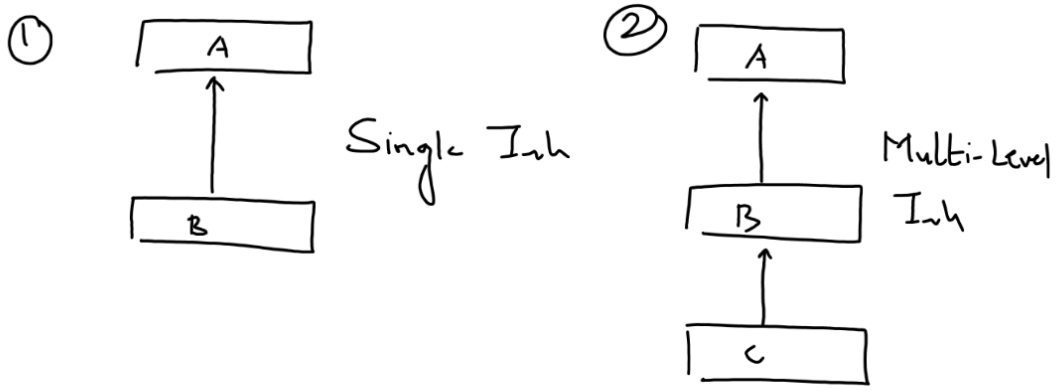
# INHERITANCE

Inheritance is the next impt pillar of OOP other than ENCAPSULATION , ABSTRACTION & POLYMORPHISM.

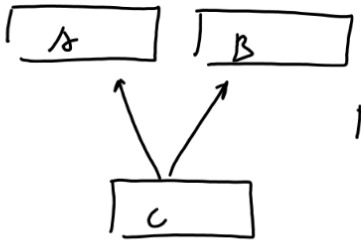
To inherit means to acquire features of an existing entity in a newly created entity. Just like a child inherits the features of his/her parents, similarly inheritance allows us to extend the features ( data & member func) of an existing class in a newly created class.

Thus we can say that INHERITANCE allows CODE REUSABILITY

## Types Of Inheritances

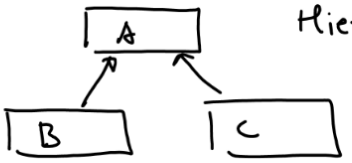


③



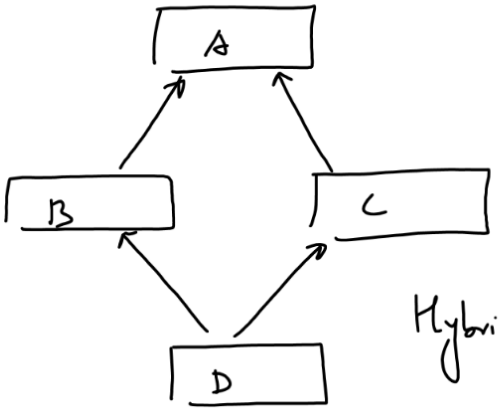

Multiple Inh

④



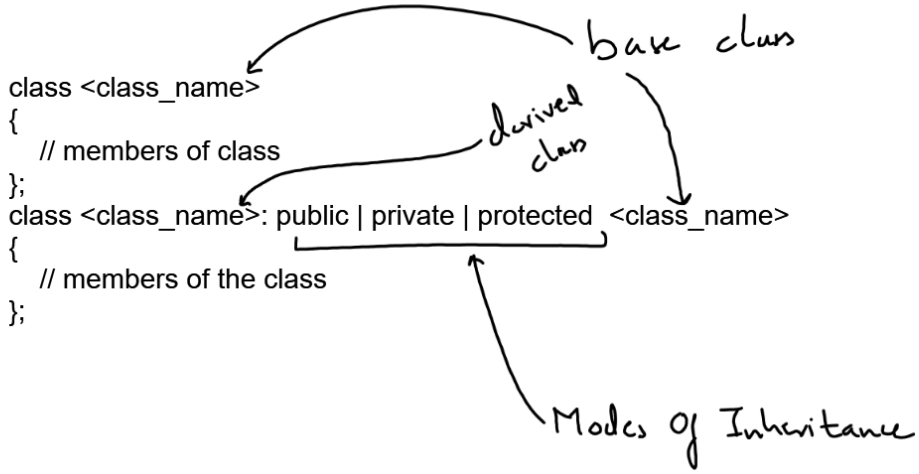

Hierarchial Inh

⑤




Hybrid Inh

**Syntax Of Inheritance In C++**




Single Inheritance In "public" Mode

```
class Box
{
  int l,b,h;
public:
        void get()
        {
          cout<<"enter l,b,h:";
          cin>>l>>b>>h;
        }
        void show()
        {
          cout<<l<<","<<b<<","<<h<<endl;
        }
};
```

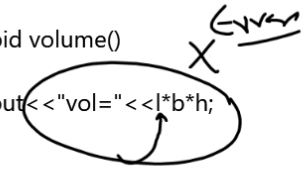

```
class Carton : public Box
{
  char mat[20];
public:
        void set()
        {
          cout<<"enter mat:";
          cin>>mat;
        }
        void display()
        {
          cout<<mat<<endl;
        }
};
```

```
int main()
{

  Carton obj;

  obj.get();

  obj.set();

  obj.show();

  obj.display();

  return 0;
}
```

## Single Inheritance In "public" Mode

```
class Box
{
   int l,b,h;
public:
        void get()
        {
          cout<<"enter l,b,h:";
          cin>>l>>b>>h;
        }
        void show()
        {
           cout<<l<<","<<b<<","<<h<<endl;
        }
};
```


```
class Carton : public Box
{
   char mat[20];
public:
        void set()
        {
          cout<<"enter mat:";
          cin>>mat;
        }
        void display()
        {
           cout<<mat<<endl;
        }
        void volume()
       {
        cout<<"vol="<<l*b*h;
        }
```



```
int main()
{

   Carton obj;

   obj.get();

   obj.set();

   obj.show();

   obj.display();

    obj.volume();
    return 0;
}
```

Why the previous code shows error ?
==========
The error is occurring because even after inheritance the private members of the base class remain private to their own class and can neither be accessed by the member functions of the derived class nor by the object of the derived class.

Solution:
======
Make the data members l,b,h as protected in the Box class.

What is protected ?
==========
1. protected is an access specifier just like private and public as well as a keyword also.

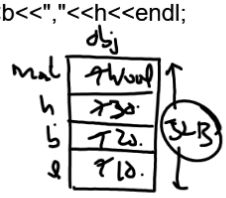
2. protected members of a class behave same as private without inheritance. That is they cannot be accessed directly from any non member function or main() function.

3. But when we inherit a class protected becomes accessible in the member functions of the derived class while private members of the base class still remian inaccessible in the derived class member functions also.

4. Thus protected is always used w.r.t inheritance and is also called as SEMI PRIVATE or INHERITANCE READY members

# Single Inheritance In "public" Mode

```
class Box
{
 protected:
   int l,b,h;
public:
        void get()
        {
          cout<<"enter l,b,h:";
          cin>>l>>b>>h;
        }
        void show()
        {
           cout<<l<<","<<b<<","<<h<<endl;
        }
};
```




```
class Carton : public Box
{
   char mat[20];
public:
        void set()
        {
          cout<<"enter mat:";
          cin>>mat;
        }
        void display()
        {
           cout<<mat<<endl;
        }
        void volume()
        {
          cout<<"vol="<<l*b*h;
        }
```

will work because l,b,h are protected

```
int main()
{

   Carton obj;

   obj.get();

   obj.set();

   obj.show();

   obj.display();

   obj.volume();
   return 0;
}
```