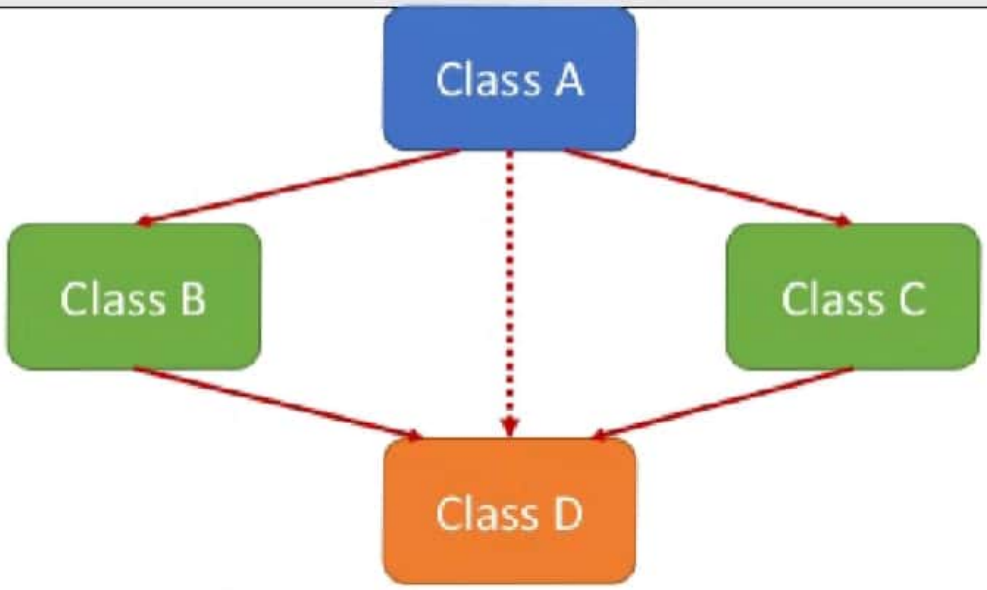
Hybrid Inheritance and Virtual Class

- **What is a virtual base class?**
- An ambiguity can arise when several paths exist to a class from the same base class. This means that a child class could have duplicate sets of members inherited from a single base class.
- C++ solves this issue by introducing a virtual base class. When a class is made virtual, necessary care is taken so that the duplication is avoided regardless of the number of paths that exist to the child class.

Hybrid Inheritance and Virtual Class

- In Multiple Inheritance, the derived class inherits from more than one base class. Hence, in Multiple Inheritance there are a lot chances of ambiguity.
- Virtual base class is used in situation where a derived have multiple copies of base class. Consider the following figure:



Hybrid Inheritance and Virtual Class

```
class A
{
        void show();
};
class B:public A {};

class C:public A {};

class D:public B, public C {};

int main()
{
 D obj;
 obj.show();
}
```

Hybrid Inheritance and Virtual Class

```
class A
{
        void show();
};
class B:public A {};

class C:public A {};

class D:public B, public C {};

int main()
{
 D obj;
 obj.show();
}
```

Most Parent Class

Hybrid Inheritance and Virtual Class

```
class A
{
        void show();
};
class B:public A {};

class C:public A {};

class D:public B, public C {};

int main()
{
 D obj;
 obj.show();
}
```

Class D inherit the Properties of both Class B and Class C. Hence class D has two inherited copies of function show().(Member Function)

Hybrid Inheritance and Virtual Class

```
class A
{
        void show();
};
class B:public A {};

class C:public A {};

class D:public B, public C {};

int main()
{
 D obj;
 obj.show();
}
```

In main() function when we call function show(), then ambiguity arises, because compiler doesn't know which show() function to call. Hence we use Virtual keyword while inheriting class.

Hybrid Inheritance and Virtual Class

- Now by adding virtual keyword, we tell compiler to call any one out of the two show() functions.

```
class B : virtual public A {};

class C : virtual public A {};

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

Hybrid Inheritance and Virtual Class

- Now by adding virtual keyword, we tell compiler to call any one out of the two show() functions.

Use of Virtual Keyword

```
class B : virtual public A {};

class C : virtual public A {};

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

Syntax use of Virtual Keyword

```
class A {
        .....
        .....
};
class B : virtual public A  {
        .....
        .....
};
class C : virtual public A {
        .....
        .....
};
class D : public B, public C {
        ....// only one copy of A
        ....// will be inherited
};
```

Virtual Class

- **What is Virtual base class? Explain its uses.**
- When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class. This can be achieved by preceding the base class' name with the word virtual.

Virtual base class Example-1

```
class A {
  public:
     int i;  //Member Variable of Class A
};
class B : virtual public A {
  public:
     int j;
};
class C: virtual public A {
  public:
     int k;
};
class D: public B, public C {
  public:
     int sum;
};
```

Virtual base class Example-1

```
class A {
  public:
    int i;  //Member Variable of Class A
};
class B : virtual public A {
  public:
    int j;
};
class C: virtual public A {
  public:
    int k;
};
class D: public B, public C {
  public:
    int sum;
};
```

Parent Class

Class B inherit the Member Variable of Parent Class A

Class C also Inherit the Member Variable of Parent Class A

Class D inherit the Member Variables of Both Class B and D

Virtual base class Example-1

```
class A {
  public:
    int i;  //Member Variable of Class A
};
class B : virtual public A {
  public:
    int j;
};
class C: virtual public A {
  public:
    int k;
};
class D: public B, public C {
  public:
    int sum;
};
```

After Inheritance D class Look like this

```
class D: public B, public C {
  public:
    int i;
    int j;
    int k;
    int sum;
};
```

Note above only one copy of Member Variable of "**int i**" is inherited due to virtual keyword

Virtual base class Example-2

```
int main()
{
  D ob;
  ob.i = 10;
  ob.j = 20;
  ob.k = 30;
  ob.sum = ob.i + ob.j + ob.k;

  cout << "Value of i is : "<< ob.i<<endl;
  cout << "Value of j is : "<< ob.j<<endl;
  cout << "Value of k is : "<< ob.k<<endl;
  cout << "Sum is : "<< ob.sum <<endl;

  return 0;
}
```

```
Value of i is : 10
Value of j is : 20
Value of k is : 30
Sum is : 60

--------------------------------
Process exited after 0.01504 seconds with return value 0
Press any key to continue . . .
```