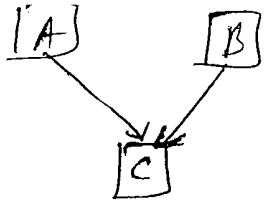
(3) Multiple inheritance

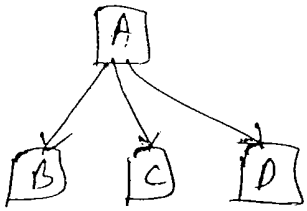



```
class A
{
  ---
  ---
};

class B
{
  ---
  ---
};

class C : public A, public B
{
  ---
  ---
};
```

(4) Hierarichal inheritance




```
class A
{
  ---
  ---
};

class B : public A
{
  ---
  ---
};

class C : public A
{
  ---
  ---
};

class D : public A
{
  ---
  ---
};
```

Which creates a pointer of type Demo class.

Now we can store address of this object into pointer as

```
P = &d ;
```

Now any data member or function of Demo class can be accessed using pointer as :-

```
P -> show( ) ;
```


Note :- We can also create objects dynamically and can store the address into pointers as-

→ `Demo *p = new Demo ;`

→
```
Demo *t ;
t = new Demo ;
```

```
void main()
{
    Vehicle *p;
    Vehicle t;
    Car c;
    p = &t;
    p->show();
    p = &c;
    p->show();
    getch();
}
```

output

show

show

In the absence of virtual keyword if we have pointer of base class then regardless of address of object of what type is stored in the pointer base class version of the function is called.

When virtual keyword is present, function is called by checking the ~~contents~~ which type of object address is hold by pointer not check which type of object ~~address~~ is created.

here the compulsory for the derived class to override the pure virtual function of base class.

```
class Vehicle
{
public:
virtual void show() = 0;
};
class Car : public Vehicle
{
public:
void show()
{
cout << "show";
}
};
void main()
{
Vehicle *p = new Car;
p -> show();
getch();
}
```

Output

show