INHERITANCE

Q) What is Inheritance? (Inh.)

A Inh. is the property by which one class acquires methods (variables) & attributes (fun) of another class.

- In inh. derived class acquires all protected & public members of a base class.

Q) Why is it done?
What is the need of inh.?
Uses / Purpose of inh.
Advantages of inh.

A 1) Reusability :-
Code of base class can be reused by derived class w/o making any changes in base class.

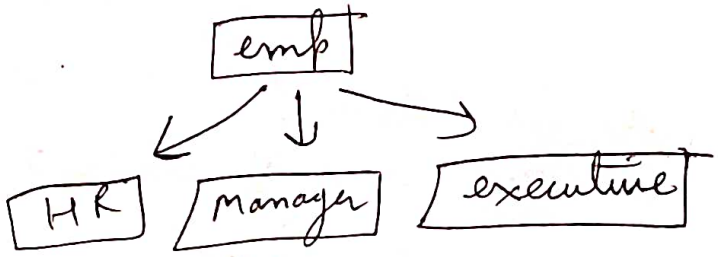
2) Extensibility :-
Base class can be extended by adding more features. For eg A simple calculator

class can be extended to create a more complex scientific-cal class with more features.

3) Creating class libraries
- For eg cout is an object of ~~ios~~ ostream class wh. is derived from ios class.

4) Making a general class more specific
- A base class is a general class & derived is specific with specialized features.

eg Employee class derived into specific classes.



TYPES OF INHERITANCE

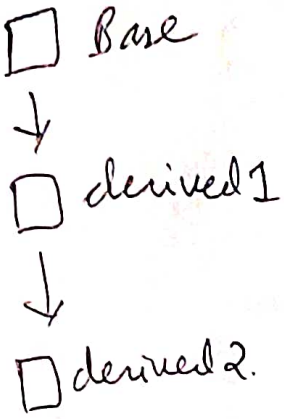
5 types of inh.

1) Single inh.

1 derived class inherits a single base class.

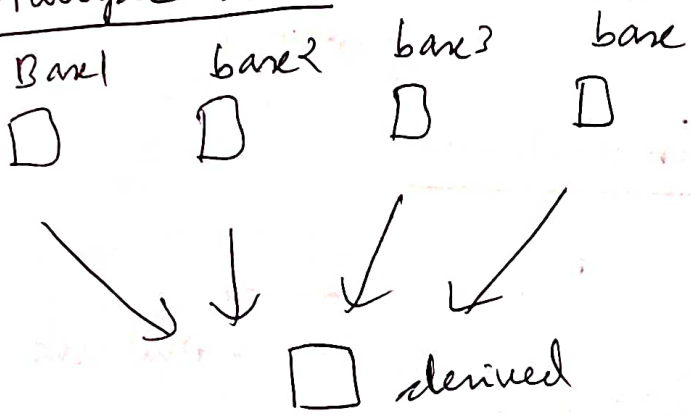
class derived : public base.

2) multi-level inh.



A derived class is further inherited by another derive class.

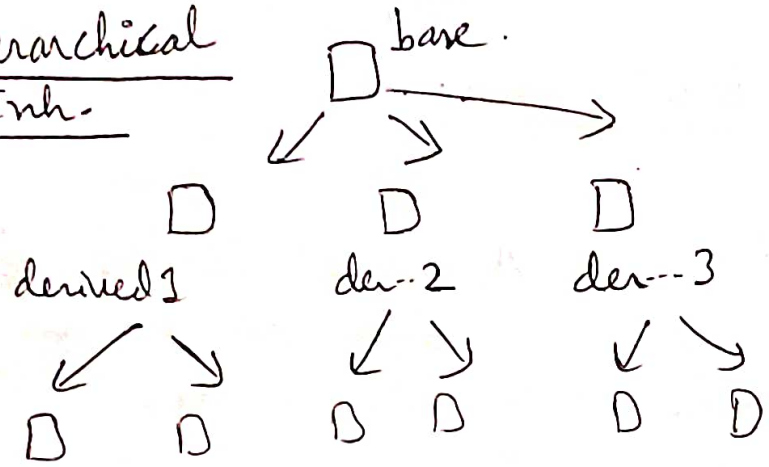
3) Multiple inh.



Syntax: class derived : public base1, public base2, public base3, public base4

A single derived class inherits from multiple base classes.

4) Hierarchical Inh.

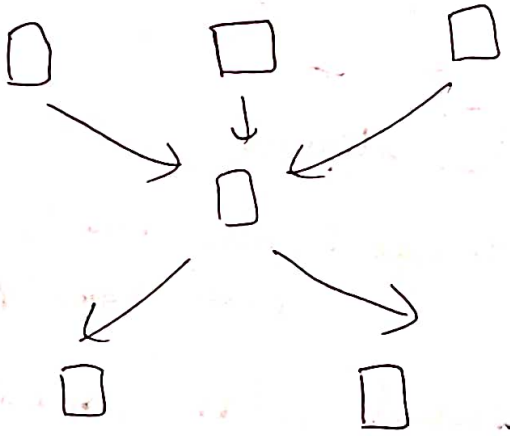


syntax:- Class derived 1 : public base

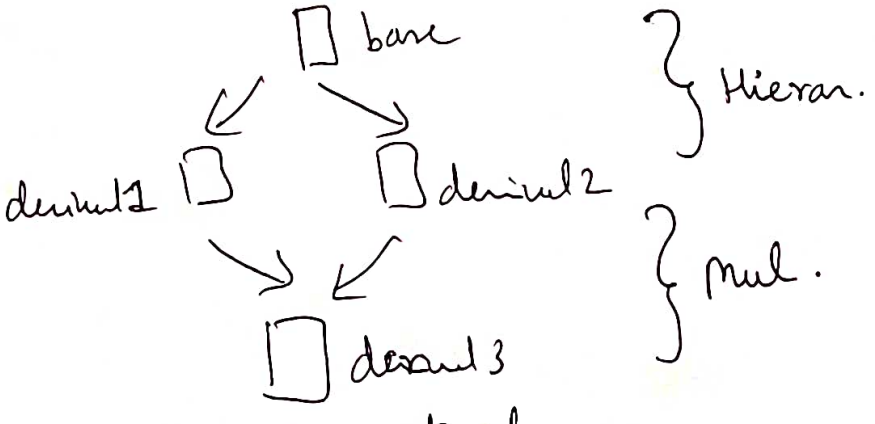
.....

A single base class inherited by mul derived classes. A tree like struct. is formed.

5) Hybrid inheritance



A combination of ~~one~~ two or more inh.



//Diamond like struct.

Protected Acces Specifier

~~• A class can~~

Q) What are acces specifiers?

A) A class can have three acces specifiers public, private, protected.

Q) ~~What~~ What does Acces specifier do?

A) They specify two properties of the members.

1) Whether obj can acces those members
2) Whether those members can be inherited.

such as:

Public :- pub. members can be accessed by obj. & pub. members can be inherited

Private :- pri members can't be accessed by obj & private members can't be inherited

Protected :- pro. members can't be accessed by obj but pro. members can be inherited

Remember: Protected can be inherited private can't.

When using inheritance, declare variables as protected instead of private. Functions will be public.

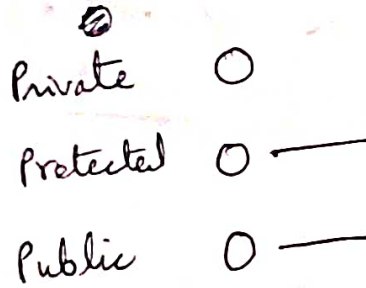
Modes of inh

- During inh. code of base class is copied to derived class. There r 2 key points regarding inh.

1) What region/members of base is copied?
2) Where are they copied in ~~to~~ derived class?

Ans-1) All the Protected & public members of base class are always inherited by derived class. This can't be changed. Private members r never inherited

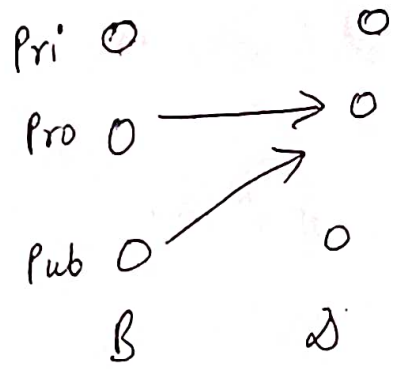
Ans-2) Where these members r copied depends on inh. modes.
There r 3 inh. modes :- public, private, protected.



In pub mode,
- Pro. members r copied to pro. region in derived class
- Pub mem r copied to pub. region in derived class.

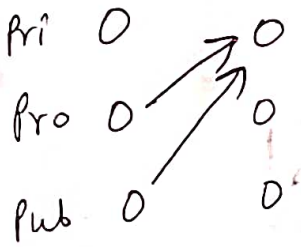
1) Public mode

Remember: Private members r never inherited.



In pro mode,
Both pro & pub members of B class r copied to protected region of D class.

2- Pro mode



In pri. mode,
pro & pub. mem of B class r copied to pri region of D class.

3- Pri mode

Note

1) Protected & public members of base class r always inherited. Pri " " " " " never "
2) Where they r copied in D class depends upon inh. mode.
3) Access specifier & inh mode r diffent concepts. Both use same key words pri, pro, pub.

- inh modes r also called Base class access specifier or inh visibility.

4) public inh mode is the most commonly used mode. It depends on your requirement.

5) During inh., declare var as protected instead of private

Syntax / structure of inh

```
class base
{
    protected:
    ==
    public:
    ==
};
class derived : public base
{
    protected:
    ==
    public:
    ==
};
```

(: → inh operator; public → inh mode; base → base class name)

Note:- base & derived classes can have any name.

Example of inh

1) W/o using inh.
Create a class wh. will add 2 var.
Create another " " " " , mul 2 var.
" " " " " , mul, div 2 var.

```
class A
{   private: //No inh, so use private.
    int i,j; sum;
    public:
    void set(int x, int y) //Fun r defined inside class to make code short.
    {
        i=x;
        j=y;
    }
    void add()
    {
        ~~cout<<i+j;~~ //add 2 var.
        sum=i+j;
    }
};
class B
{
    private:
    int i, j, sum, prod;
    public:
    void set(int x, int y)
    {
        i=x;
        j=y;
    }
```

```
void add()
}
    cout << i+j;   //add
        sum = i+j;
}
void mul()
}
    cout << i*j;   //mul.
        prod = i*j;
}
};
class C
}
    private:
    int i, j, sum, prod, quo;
    public:
    void set(int x, int y)
    }   i = x; j = y
    }
    void add()
    }
        cout << i+j;   //add
            sum = i+j;
    }
    void mul()
    }
        cout << i*j;   //mul
            prod = i*j;
    }
    void div()
    }
        cout << i/j;   //div.
        }   quo = i/j
};
```

```
int main()
{
    C obj-c; // create obj-c;
    obj-c.set(5,6);
    obj-c.add();
    obj-c.mul();
    obj-c.div();
}
```

3.

// In this eg., 3 classes r created.
Now code of class C can be reduced by using inh as shown in next eg.

Code-2 using inh.

1). Create a base class. In this, create a fun to add 2 var.
- Derive this class & create fun to add, mul 2 var.
- Further derives this class & create a fun to add, mul, div 2 var.

struct should be :-

base
↓
derived1 (der1)
↓
der2.

```
class base
{
    protected:  // use protected not pri
    int i, j, sum;
    public:
    void set(int x, int y)
    {   i = x; j = y;
    }
    void add()
    {   sum = i + j;
    }
};
class der1 : public base
{
    protected:
    int prod; // i, j, sum r derived.
    public:
    void mul()  // set & add fun r derived.
    {   prod = i * j;
    }
};
class der2 : public der1
{   protected:
    int quo; // i, j, sum, prod r derd.
    public:
    void div();
    {   quo = i / j;
    }
};
```

```
int main()
{
    der2 obj_d2;
    //create obj of der2;
    obj_d2.set(1,2);
    obj_d2.add();
    obj_d2.mul();
    obj_d2.div();
}
```

Imp points

Q) Why 3 diff classes r created? Why not add, mul, div in same class?

A) evolution. When base class is created the programmer has not thought of all fun.

- Later when prog^r wants to add more fun to base class, he can add add feat. thru inh. Simply derive base & add more fun.
- This way classes r evolved & more feat. r added.

Q2) Benefit of inh?

A Code Reusability. Compare codes of class C w/o inh & class d2 w/ inh. Both have same feat. but code of class d2 is more compact due to inh.

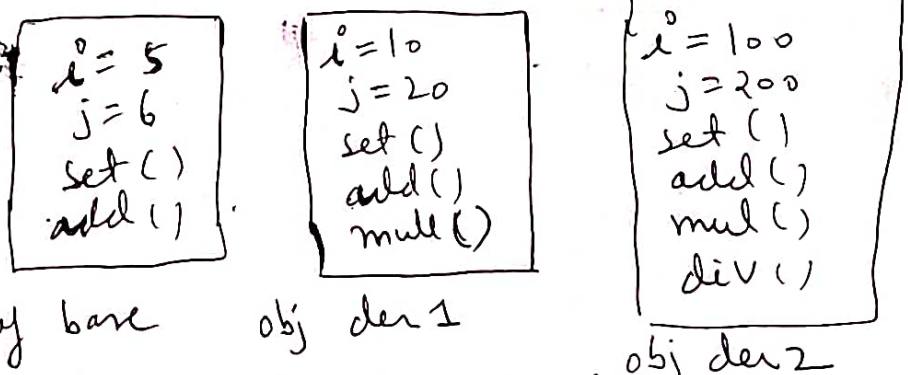
Q3) How can der2 class add, mul, div var = i & j of base class?

A) Bcoz der2 class has inherited these var. So der2 will use same var. Do not create these var again.

Q4) Will the values of i, j remain same in all class?

A) No. i & j r common in all classes but their values will be different bcoz values r not stored in class.

- Values of i, j r stored in obj. Each class will have separate obj, wh. contain diff values.



obj of base obj der 1 obj der 2

example of obj of each class

Q5) Can we create obj of base class?

A) Yes, obj of any class can be created & used.

Q6) Why should we prefer creating obj of der2 class?

A) Bcoz der2 class has max no. of feat/fun. So create obj of derived class.

obj of base can only add.

obj of der1 " add, mul

" " der2 " " " , div.

Remember: Code of der2 class is the shortest, but it has most feat.