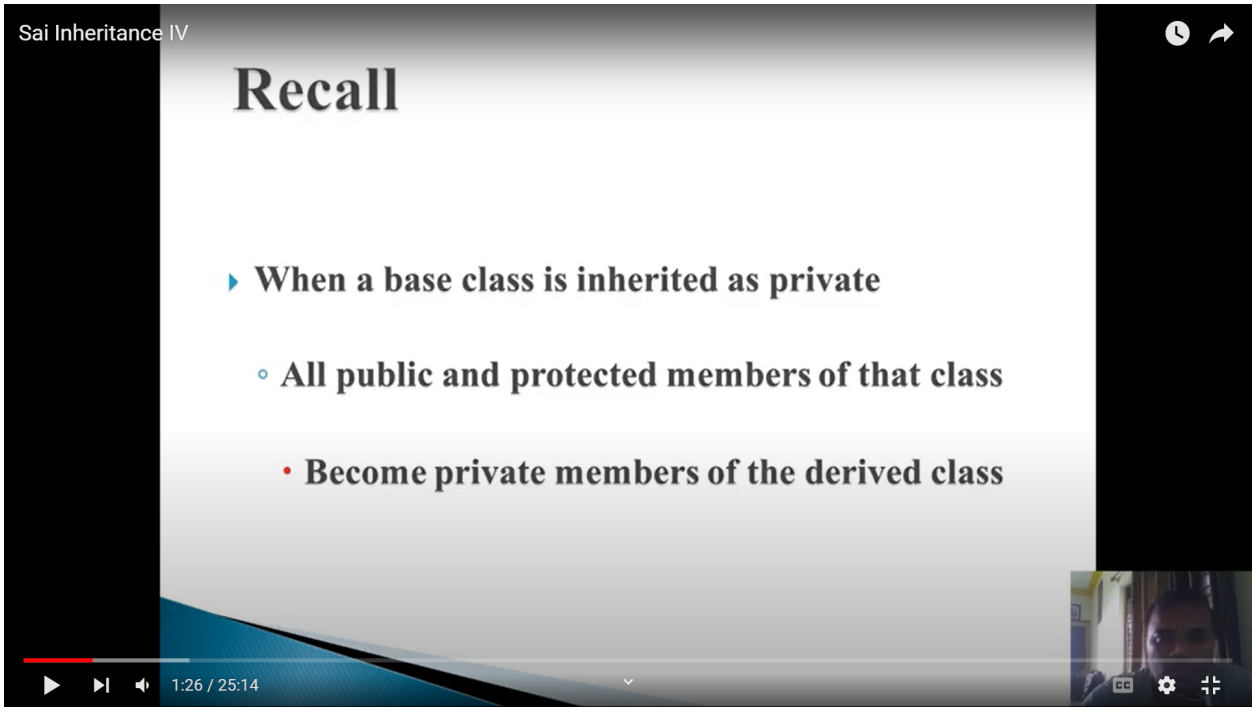
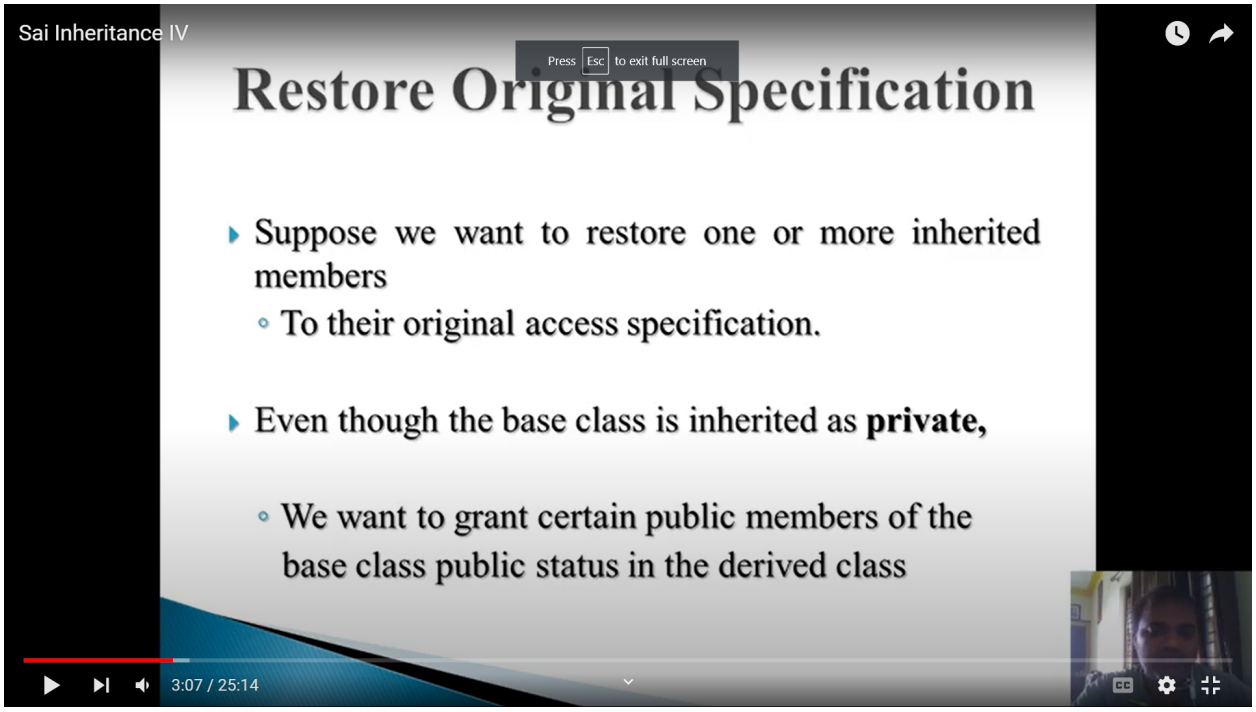
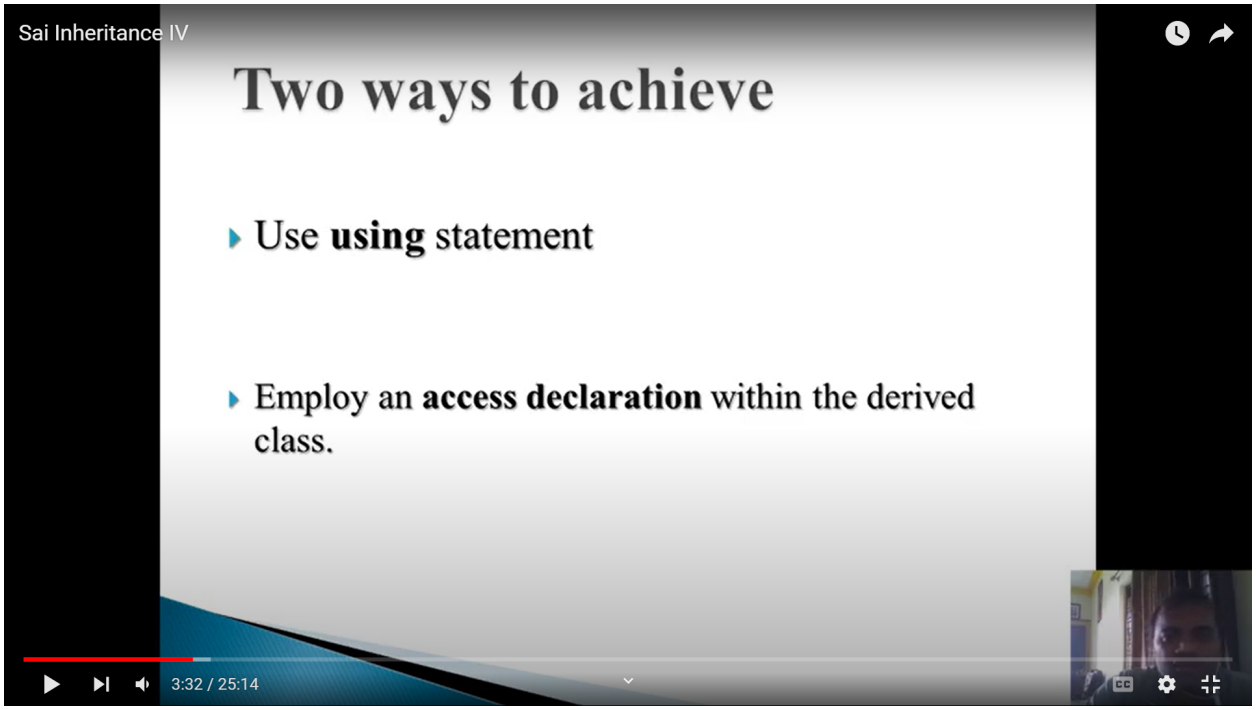
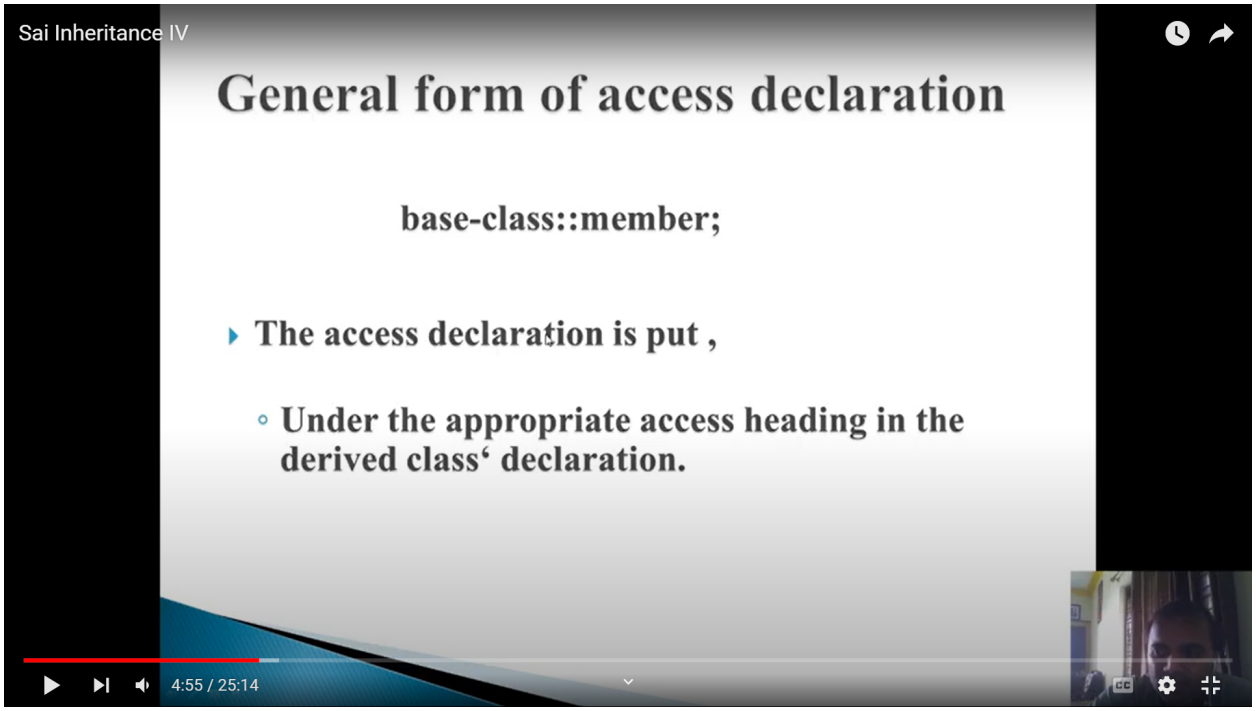
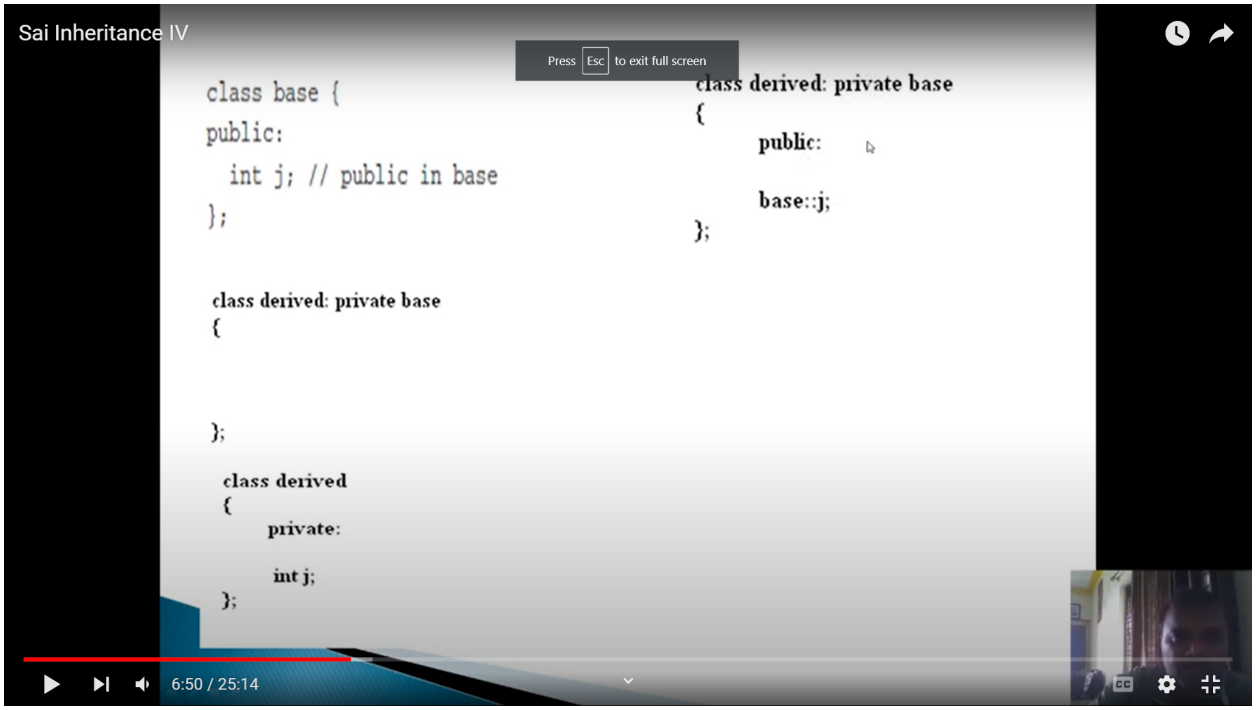
Protected = Same as private but derived classes can also access

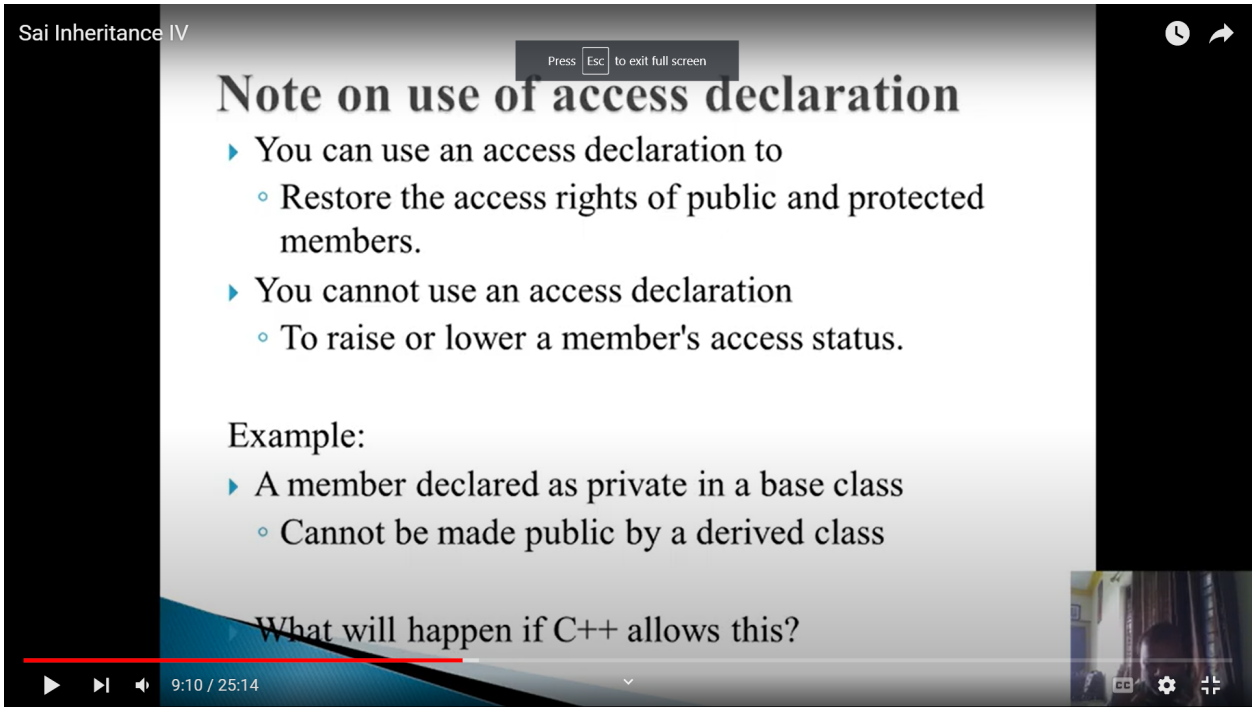





Sai Inheritance IV
Two ways to achieve
Use using statement
Employ an access declaration within the derived class.
3:32 / 25:14


Sai Inheritance IV
General form of access declaration
base-class::member;
The access declaration is put ,
Under the appropriate access heading in the derived class‘ declaration.
4:55 / 25:14


Sai Inheritance IV
Press Esc to exit full screen
class base {
public:
int j; // public in base
};
class derived: private base
{
public:
base::j;
};
class derived: private base
{
};
class derived
{
private:
int j;
};
6:50 / 25:14

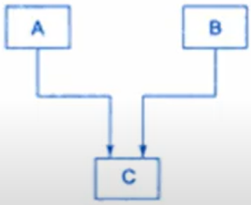

Sai Inheritance IV
Press Esc to exit full screen
Note on use of access declaration
You can use an access declaration to
Restore the access rights of public and protected members.
You cannot use an access declaration
To raise or lower a member's access status.
Example:
A member declared as private in a base class
Cannot be made public by a derived class
What will happen if C++ allows this?
9:10 / 25:14

Sai Inheritance IV

Press Esc to exit full screen

Ambiguity

- When multiple base classes are inherited,



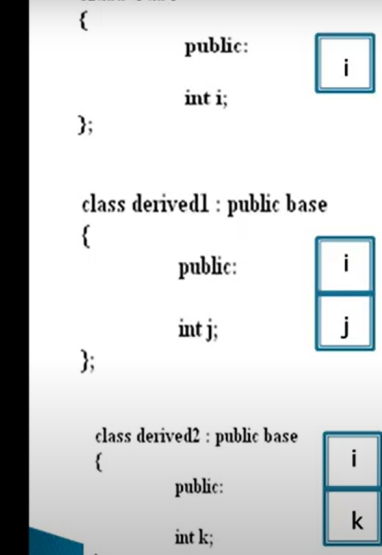
- There is chance for ambiguity.

10:09 / 25:14

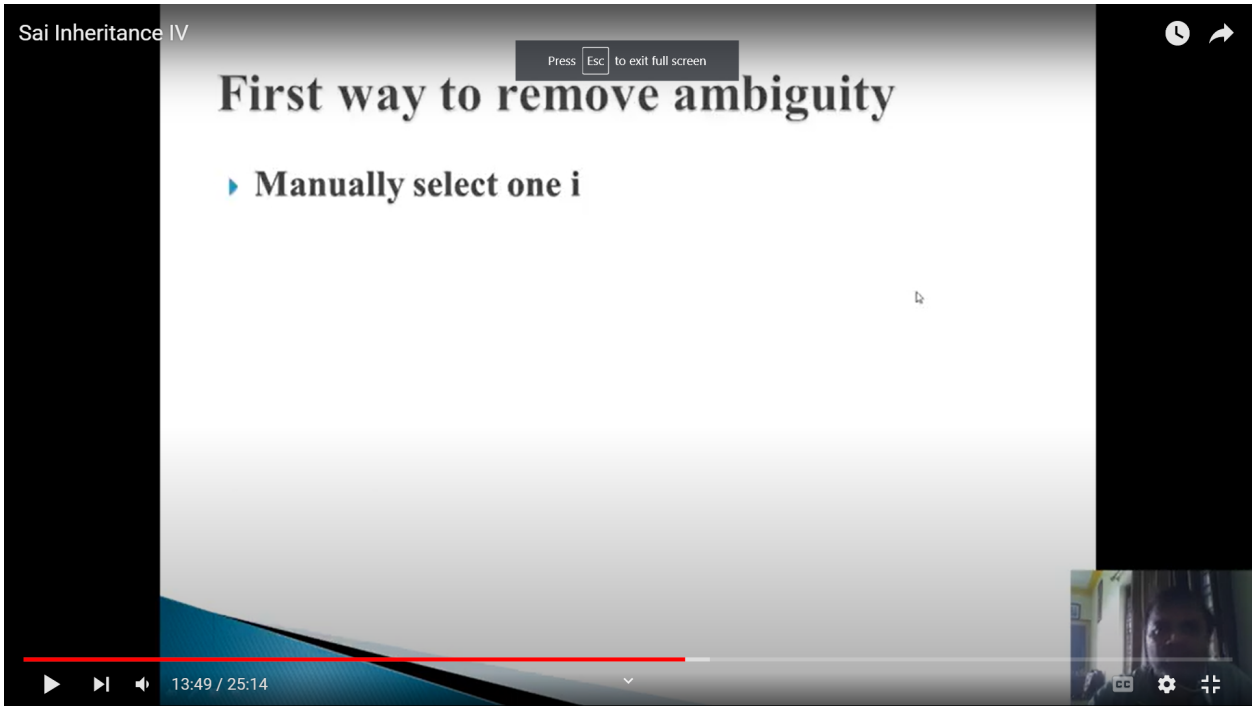
Sai Inheritance IV

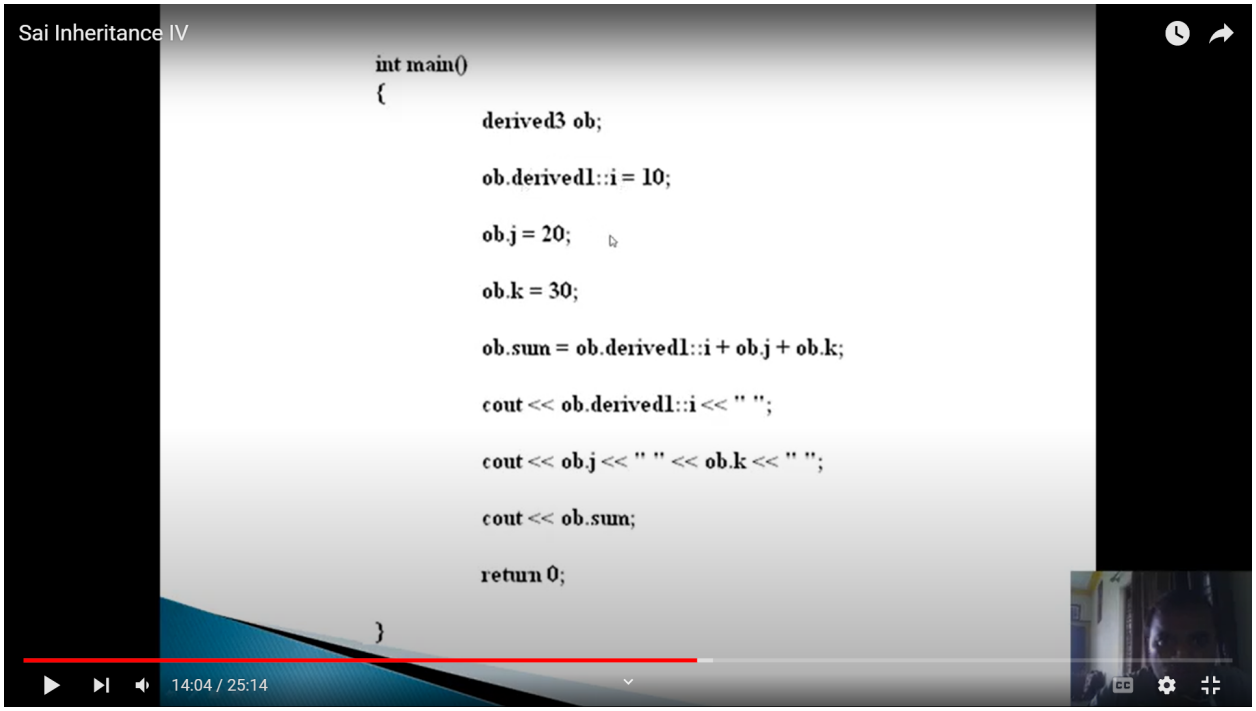
Press Esc to exit full screen

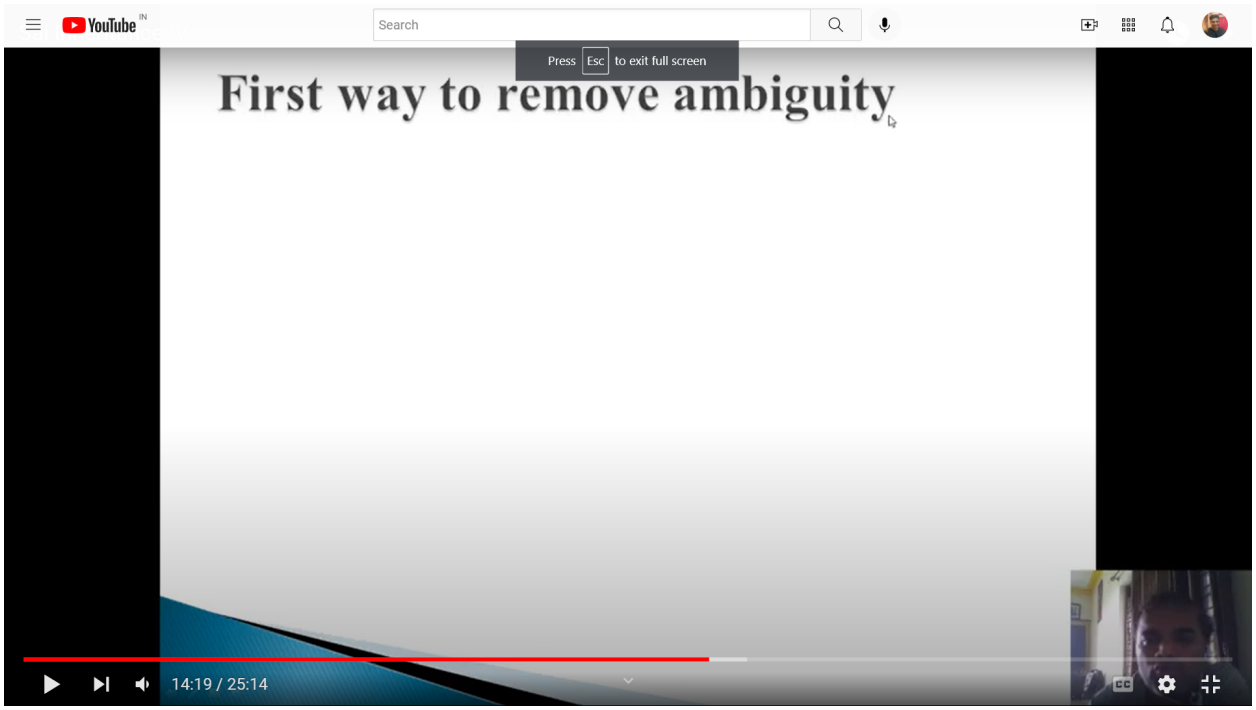
```
class base
{
        public:
        int i;
};

class derived1 : public base
{
        public:
        int j;
};

class derived2 : public base
{
        public:
        int k;
};

class derived3 : public derived1, public derived2
{
        public:
        int sum;
};
int main()
{
        derived3 ob;
        ob.i = 10;
        ob.j = 20;
        ob.k = 30;
        ob.sum = ob.i + ob.j + ob.k;
        cout << ob.i << " ";
        cout << ob.j << " " << ob.k << " ";
        cout << ob.sum;
        return 0;
}
```

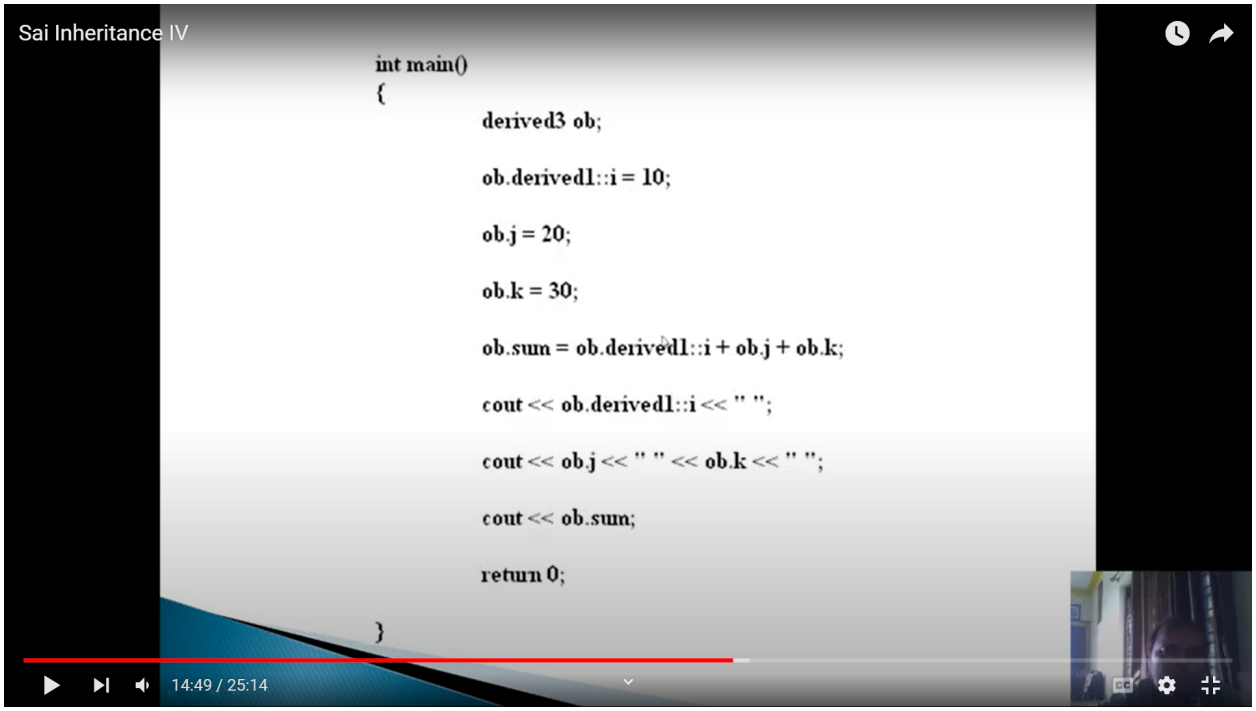


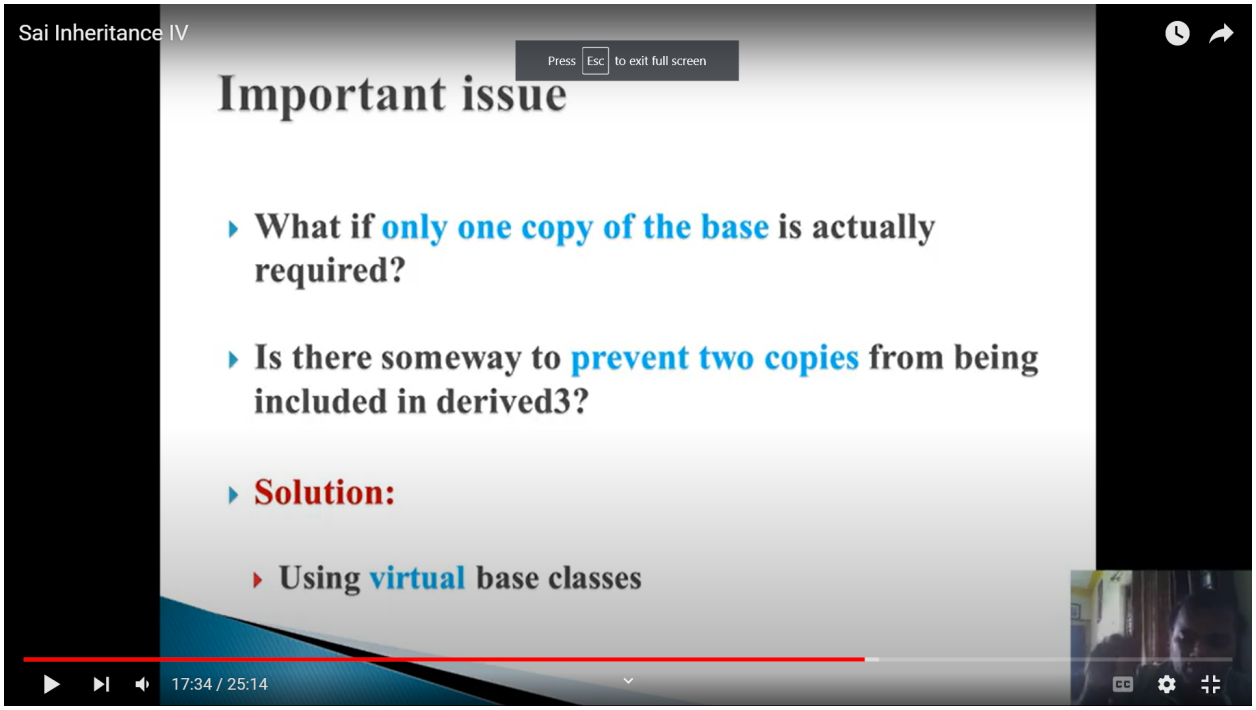
12:09 / 25:14

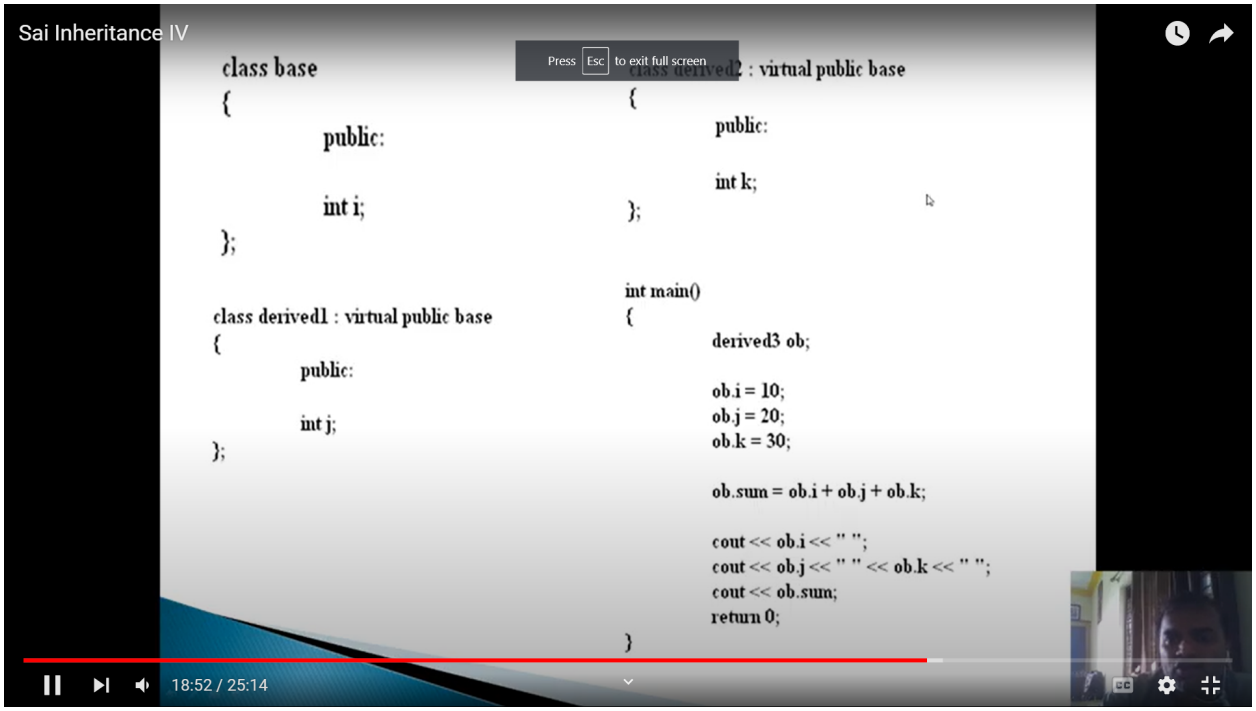

Sai Inheritance IV
Press Esc to exit full screen
First way to remove ambiguity
Manually select one i
13:49 / 25:14

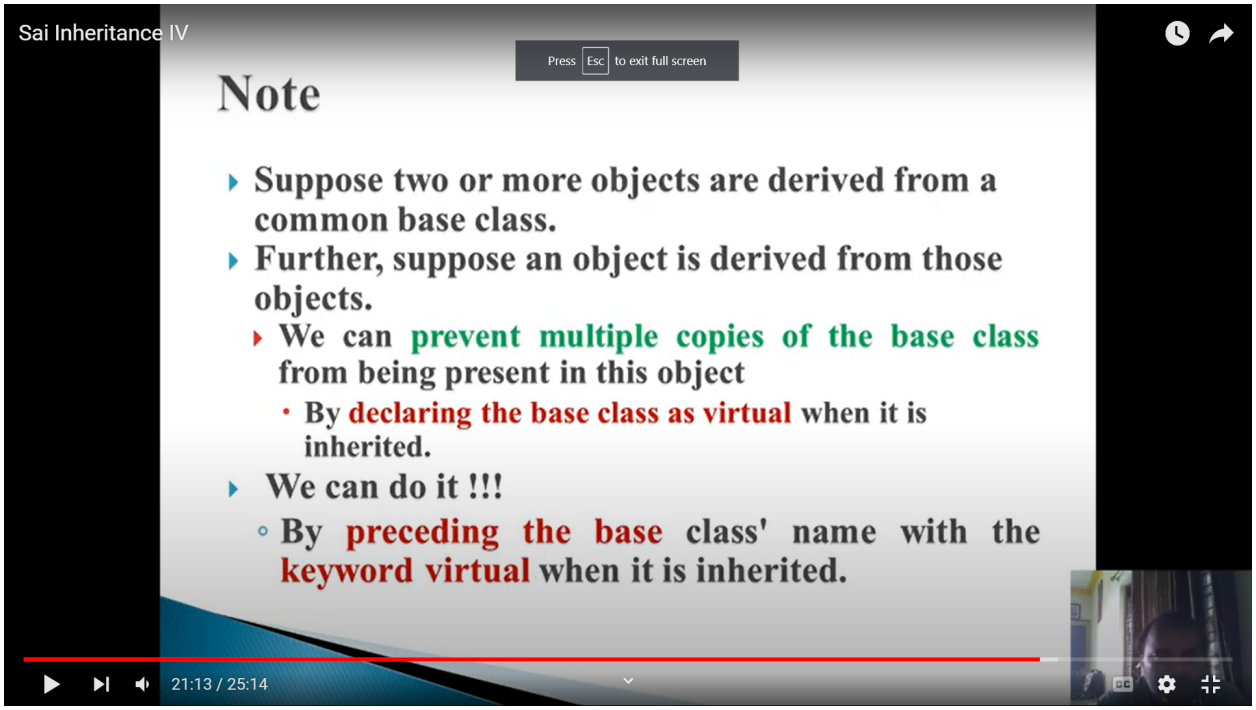

Sai Inheritance IV
int main()
{
derived3 ob;
ob.derived1::i = 10;
ob.j = 20;
ob.k = 30;
ob.sum = ob.derived1::i + ob.j + ob.k;
cout << ob.derived1::i << " ";
cout << ob.j << " " << ob.k << " ";
cout << ob.sum;
return 0;
}
14:04 / 25:14

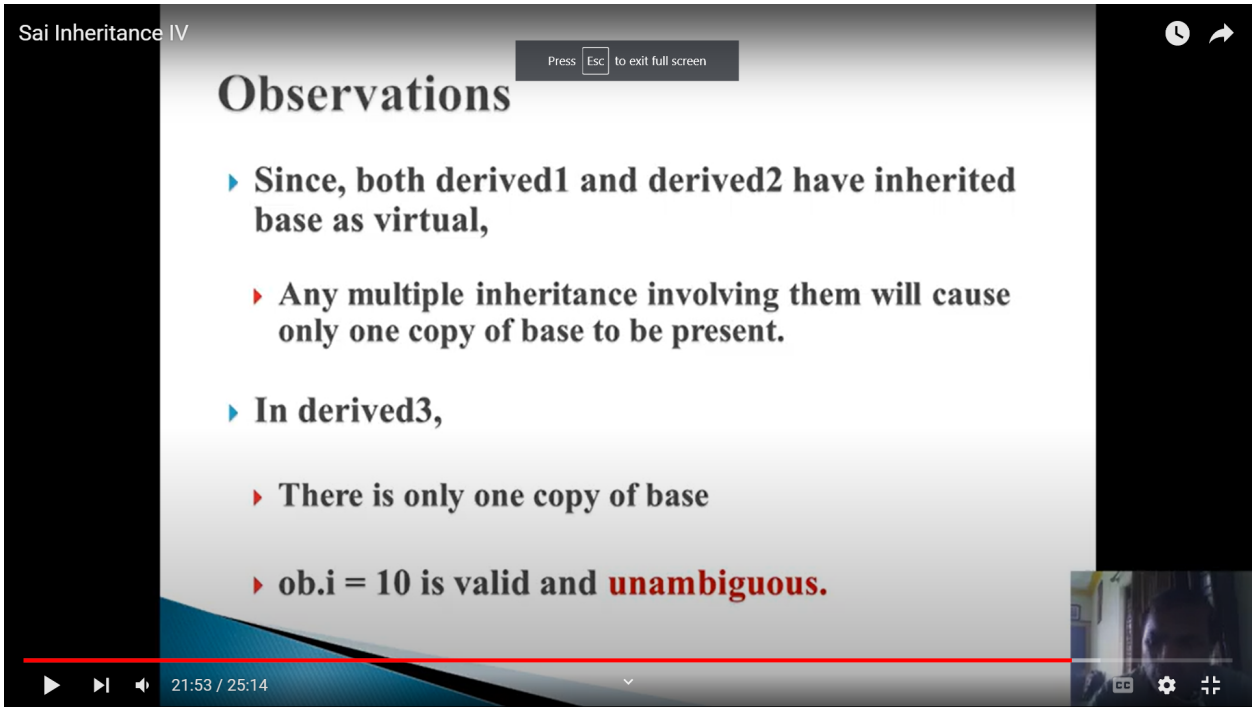

YouTube IN
Search
Press Esc to exit full screen
First way to remove ambiguity
14:19 / 25:14

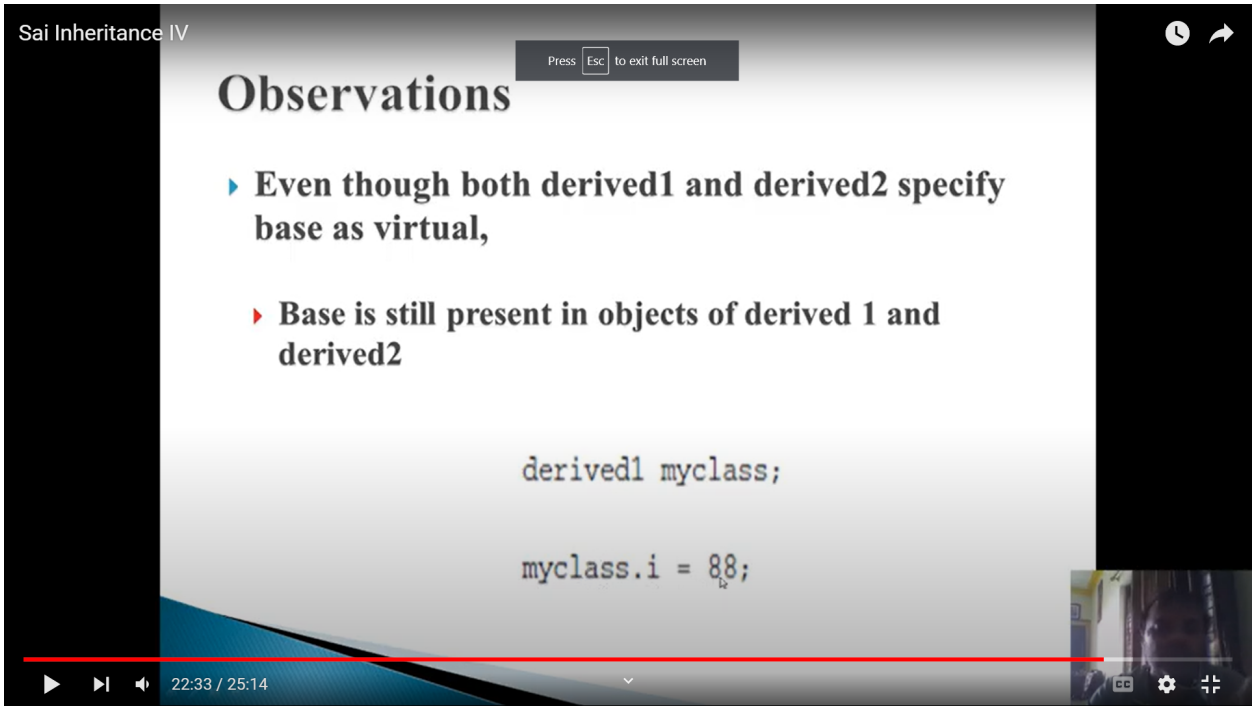

Sai Inheritance IV
int main()
{
derived3 ob;
ob.derived1::i = 10;
ob.j = 20;
ob.k = 30;
ob.sum = ob.derived1::i + ob.j + ob.k;
cout << ob.derived1::i << " ";
cout << ob.j << " " << ob.k << " ";
cout << ob.sum;
return 0;
}
14:49 / 25:14

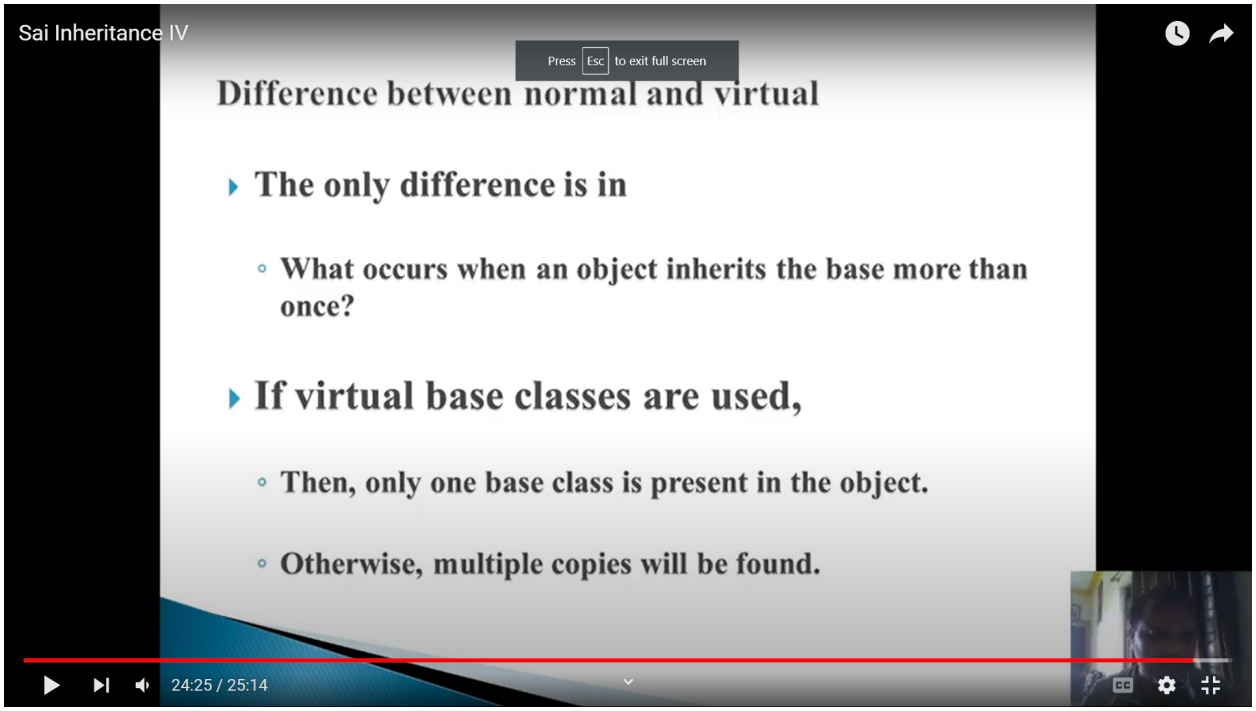

Sai Inheritance IV
Press Esc to exit full screen
Important issue
What if only one copy of the base is actually required?
Is there someway to prevent two copies from being included in derived3?
Solution:
Using virtual base classes
17:34 / 25:14


Sai Inheritance IV
Press Esc to exit full screen
class base
{
public:
int i;
};
class derived1 : virtual public base
{
public:
int j;
};
class derived2 : virtual public base
{
public:
int k;
};
int main()
{
derived3 ob;
ob.i = 10;
ob.j = 20;
ob.k = 30;
ob.sum = ob.i + ob.j + ob.k;
cout << ob.i << " ";
cout << ob.j << " " << ob.k << " ";
cout << ob.sum;
return 0;
}
18:52 / 25:14


Sai Inheritance IV
Press Esc to exit full screen
Note
Suppose two or more objects are derived from a common base class.
Further, suppose an object is derived from those objects.
We can prevent multiple copies of the base class from being present in this object
By declaring the base class as virtual when it is inherited.
We can do it !!!
By preceding the base class' name with the keyword virtual when it is inherited.
21:13 / 25:14


Sai Inheritance IV
Press Esc to exit full screen
Observations
Since, both derived1 and derived2 have inherited base as virtual,
Any multiple inheritance involving them will cause only one copy of base to be present.
In derived3,
There is only one copy of base
ob.i = 10 is valid and unambiguous.
21:53 / 25:14


Sai Inheritance IV
Press Esc to exit full screen
Observations
Even though both derived1 and derived2 specify base as virtual,
Base is still present in objects of derived 1 and derived2
derived1 myclass;
myclass.i = 88;
22:33 / 25:14


Sai Inheritance IV
Press Esc to exit full screen
Difference between normal and virtual
The only difference is in
What occurs when an object inherits the base more than once?
If virtual base classes are used,
Then, only one base class is present in the object.
Otherwise, multiple copies will be found.
24:25 / 25:14