Practice Problems on Inheritance | L:22 | C++ | Competitive Programming

Q1: when we create the child class object **then the constructor of both parent and child class** will be called.,But

a—> parent class constructor will be called first.

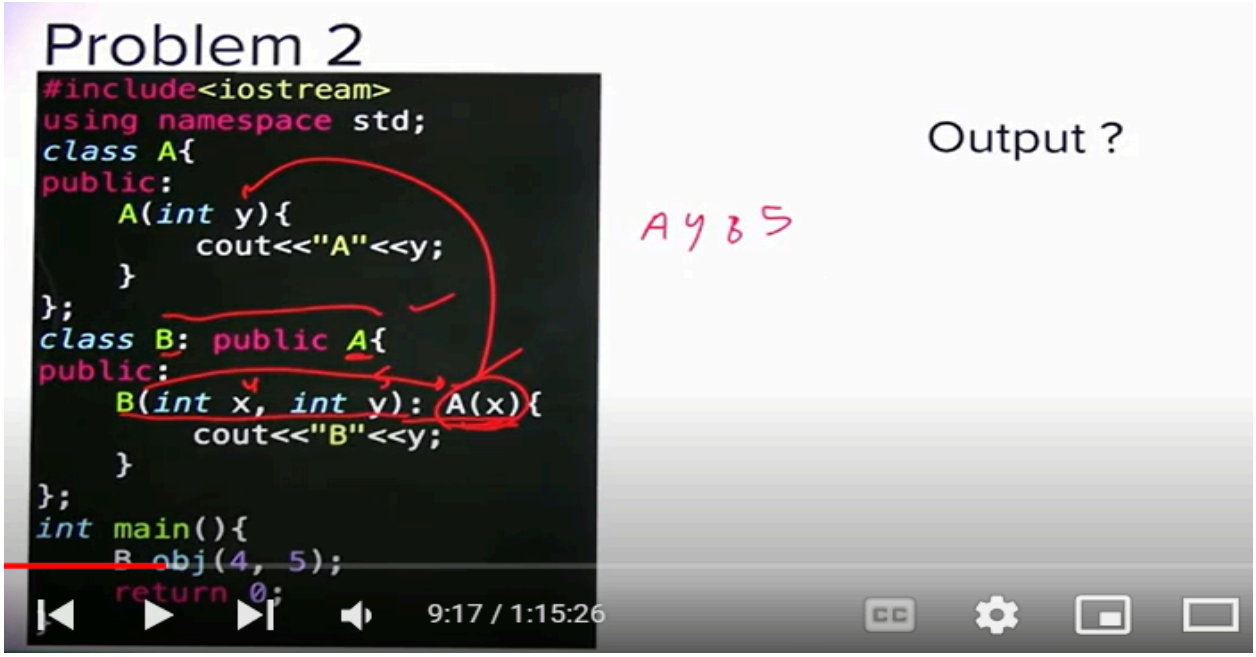
b—> then the child class constructor will be called.


```
using namespace std;
class A
{
    public:
        A()
        {
            cout<<"A";
        }
};
class B : public A
{
    public:
        B()
        {
            cout<<"B";
        }
};
class C : public B
{
    public:
        C()
        {
            cout<<"C";
        }
};
int main()
{
    C obj;
    return 0;
}
```

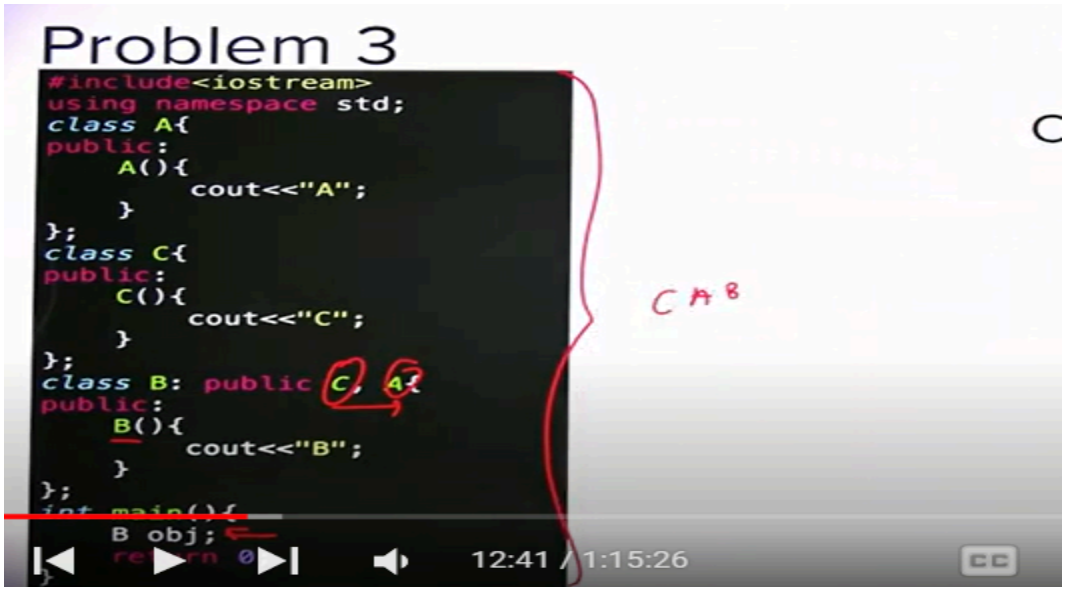
```
ABC

...Program finished with exit code 0
```

Q2: **B(int x,int y):A(x)** ⇒ way to call the parent constructor if we have to pass the argument.

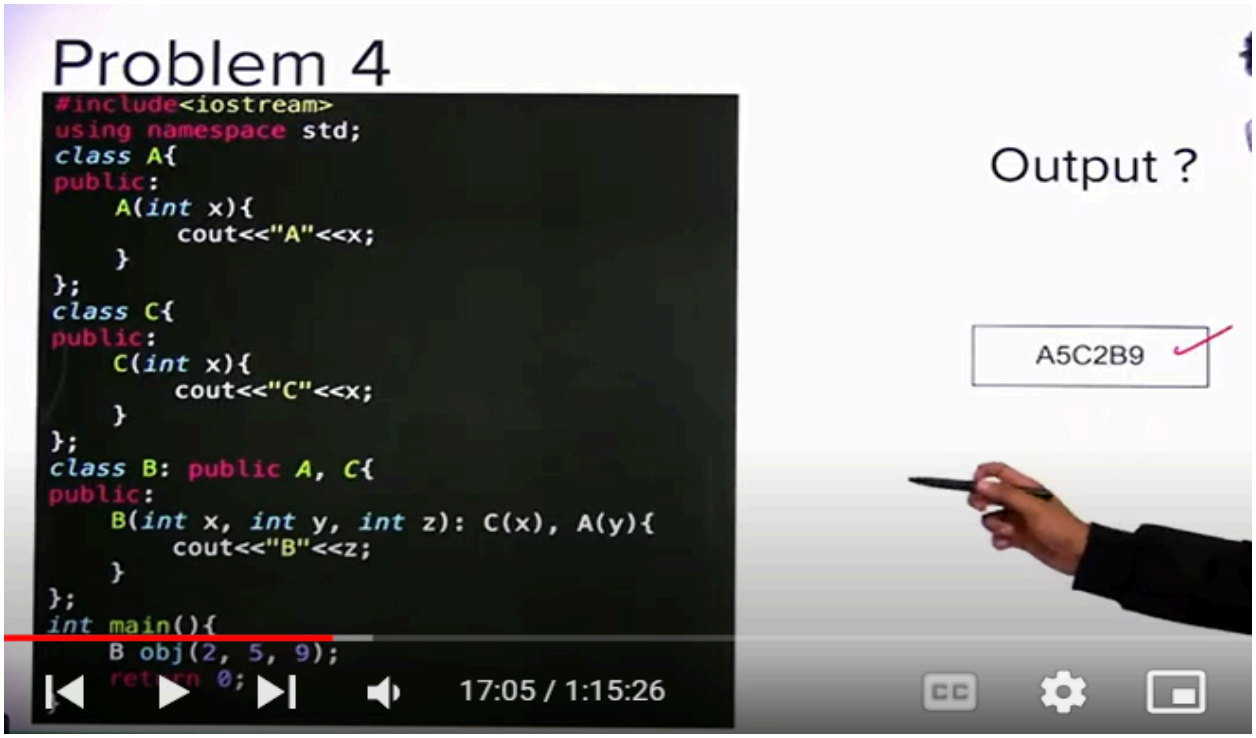


Q3: constructor are called in the order in which parent classes are written.



Q4:
constructor of parents are always called in the order in which the classes are inherited,no matter how you have written it in the constructor.

If the parent class does not have any constructor then the default constructor of the parent class will be called.

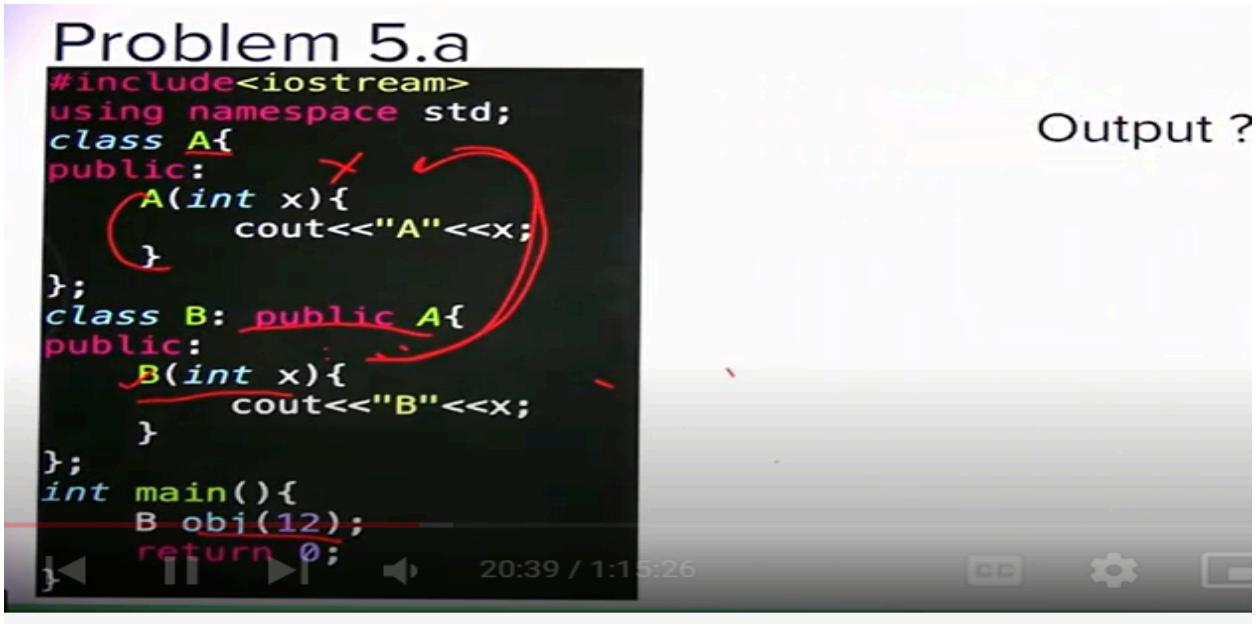


Q5-A :

1.when w**e create any constructor inside the class then the compiler will not create the default constructor.**

2. When the **obj** of the child is created , first constructor of the parent class will be called.
Here we have not called the parent class constructor explicitly in the child , then default constructor of parent class will be called.

BUt in the child default constructor,are not present and compiler also not create the default constructor bez, we have created the parameterized constructor.
So it will give **error.**



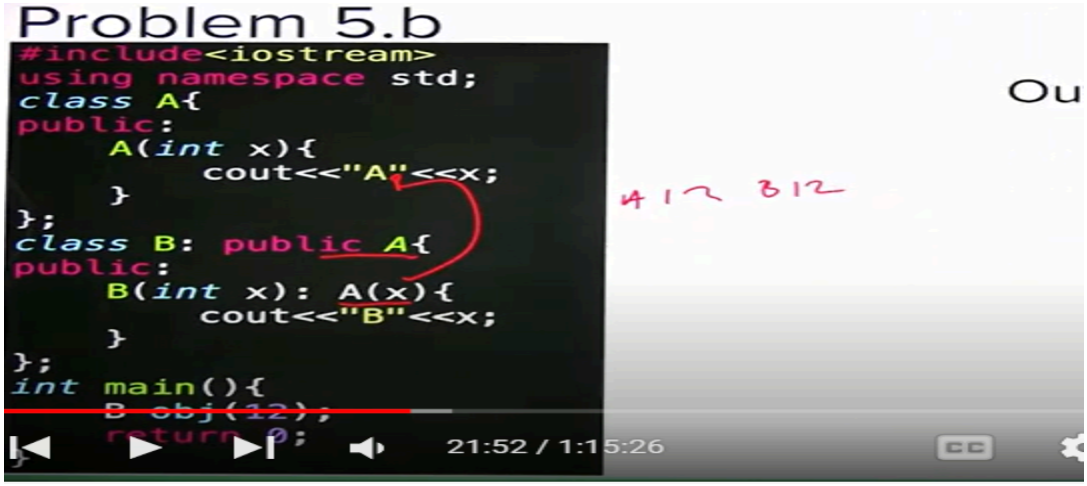
error: constructor for 'B' must explicitly initialize the base class 'A' which does not have a default constructor

```
 9  #include <iostream>
10  using namespace std;
11  class A
12  {
13      public:
14          A()
15          {
16              cout<<"A";
17          }
18  };
19  class B : public A
20  {
21      public:
22          B(int x)
23          {
24              cout<<"B"<<x;
25          }
26  };
27  int main()
28  {
29      B obj(10);
30      return 0;
31  }
32
```

```
AB10

...Program finished with exit code 0
```

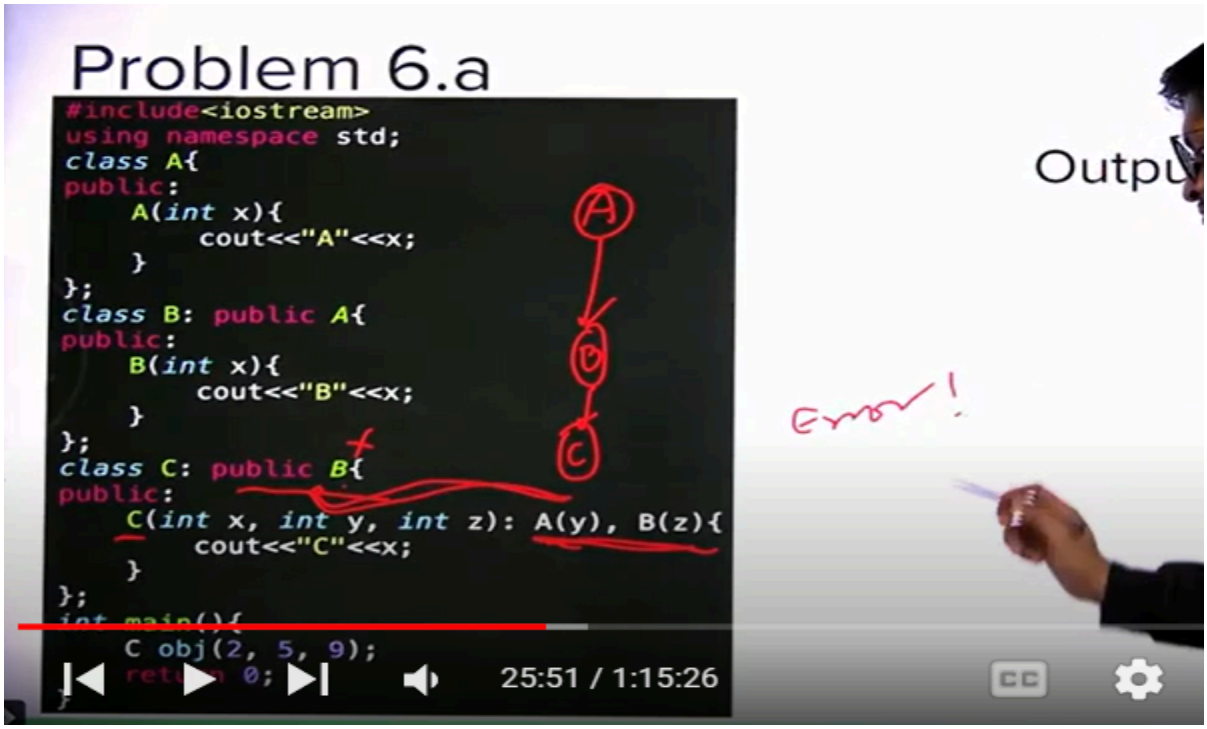
Q5-B:



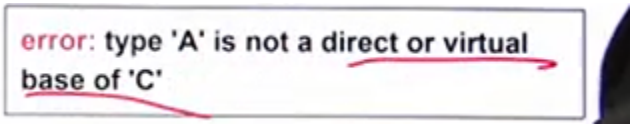
It will not give any error
Op: A 12 B 12

Q6 -A :

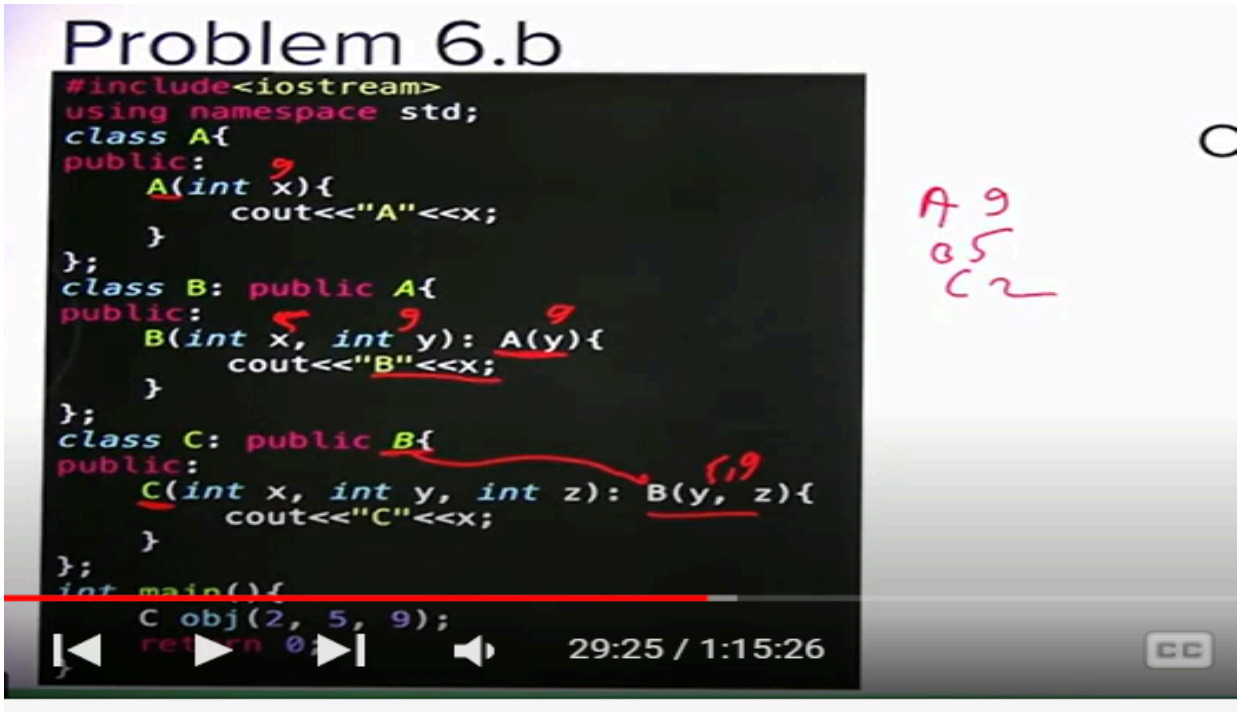
A child class can only call the constructor of its direct(immediate) parent class , not any other parent class.



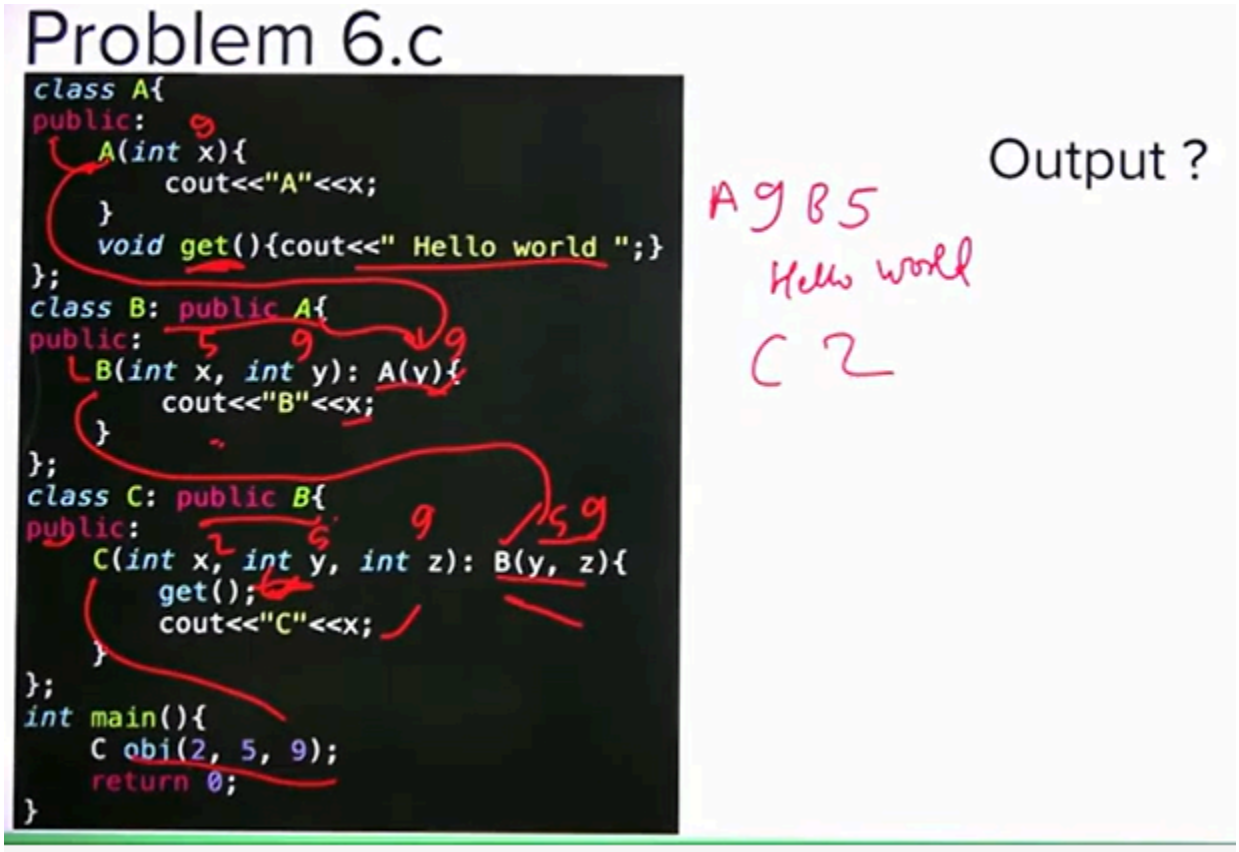
C is trying to call the constructor of class A but it is not inheriting class A.==> therefore Error



Q6-B :

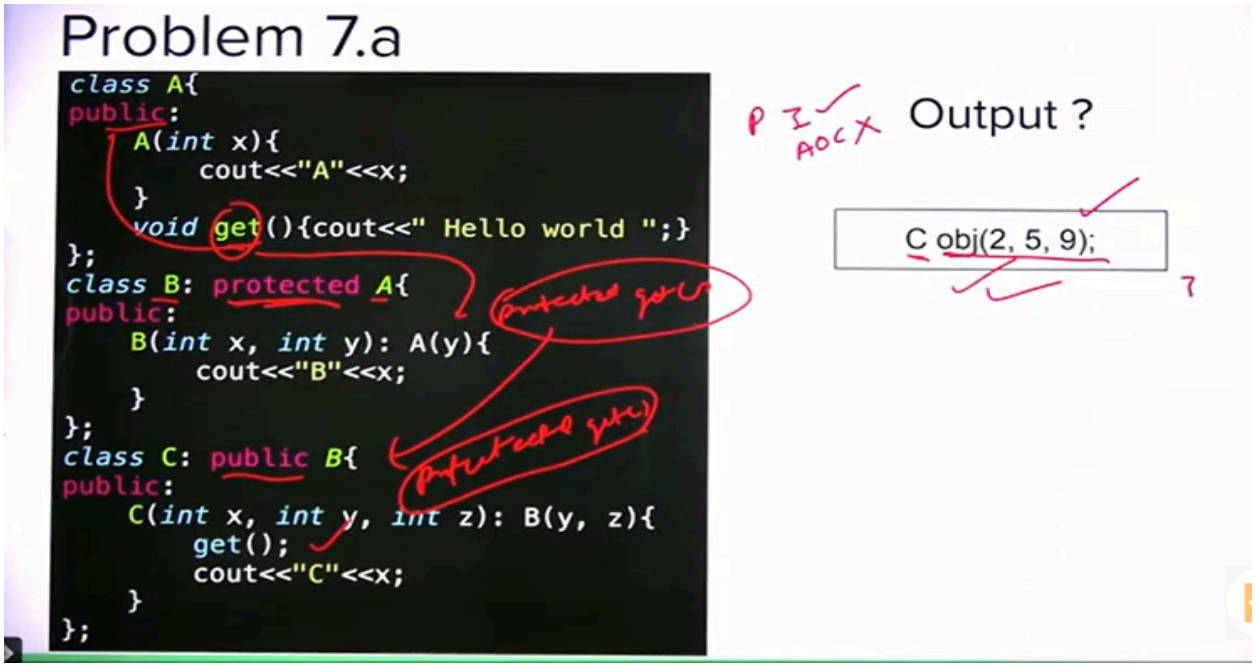

Problem 6.b
#include<iostream>
using namespace std;
class A{
public:
A(int x){
cout<<"A"<<x;
}
};
class B: public A{
public:
B(int x, int y): A(y){
cout<<"B"<<x;
}
};
class C: public B{
public:
C(int x, int y, int z): B(y, z){
cout<<"C"<<x;
}
};
int main(){
C obj(2, 5, 9);
29:25 / 1:15:26

Q6-C :

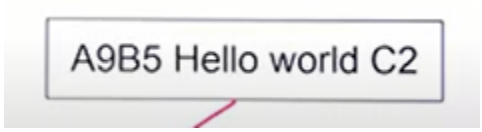

Problem 6.c
class A{
public:
A(int x){
cout<<"A"<<x;
}
void get(){cout<<" Hello world ";}
};
class B: public A{
public:
B(int x, int y): A(y){
cout<<"B"<<x;
}
};
class C: public B{
public:
C(int x, int y, int z): B(y, z){
get();
cout<<"C"<<x;
}
};
int main(){
C obj(2, 5, 9);
return 0;
}
Output ?

Q7 -A :

Protected members can be inherited.
But it can not be accessed outside of the class. Inside the class it can be accessed.

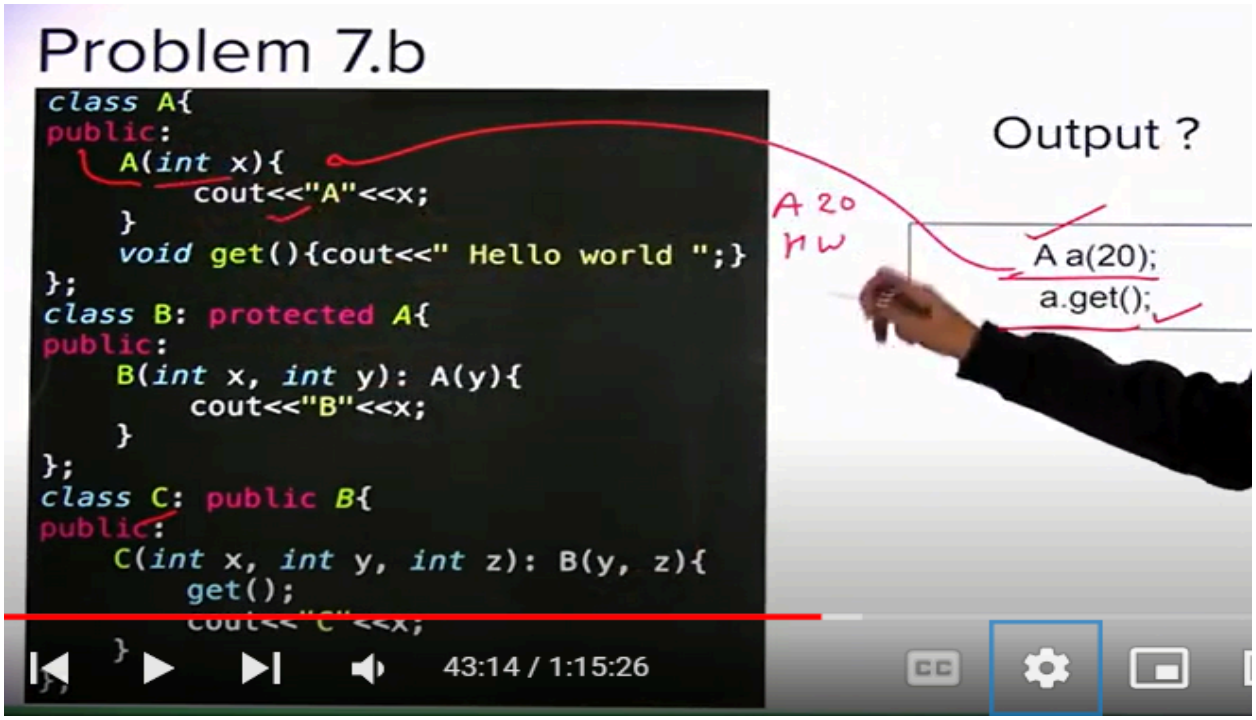


Op:

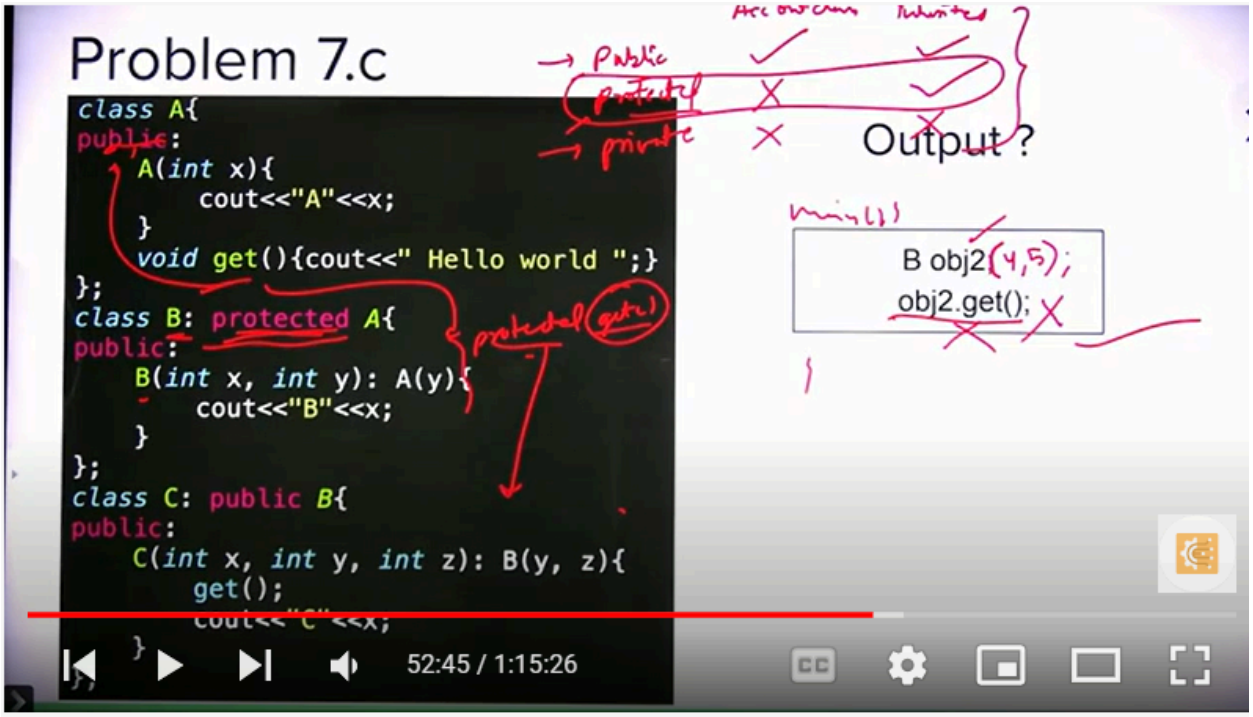


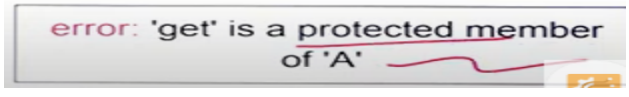
Class B is derived from the A in protected mode,so get() will become the protected in the class B.
Again class C is derived from the class B in public mode,so get() will remain protected inside the class C ,bez maximum security will be taken.

Q7 B :



Q7 C:

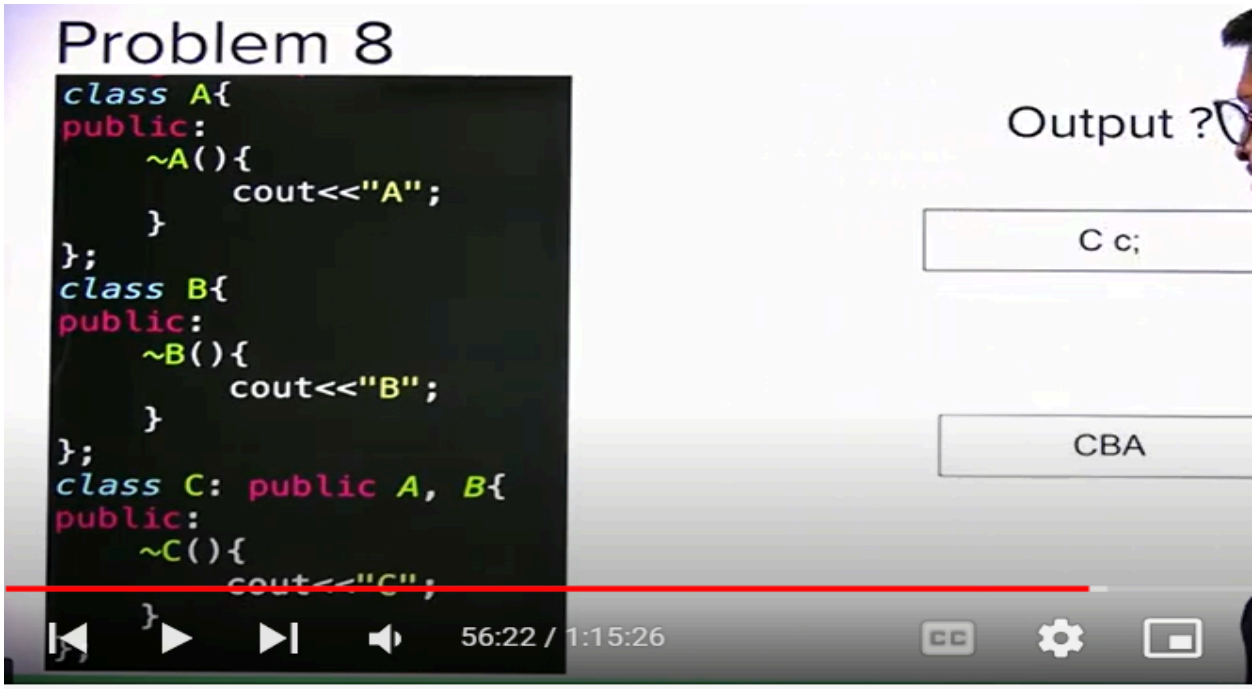




get() will be inherited but it will become protected in the class B.
Protected members can not be accessed outside of the class ===> therefore it will give the error.

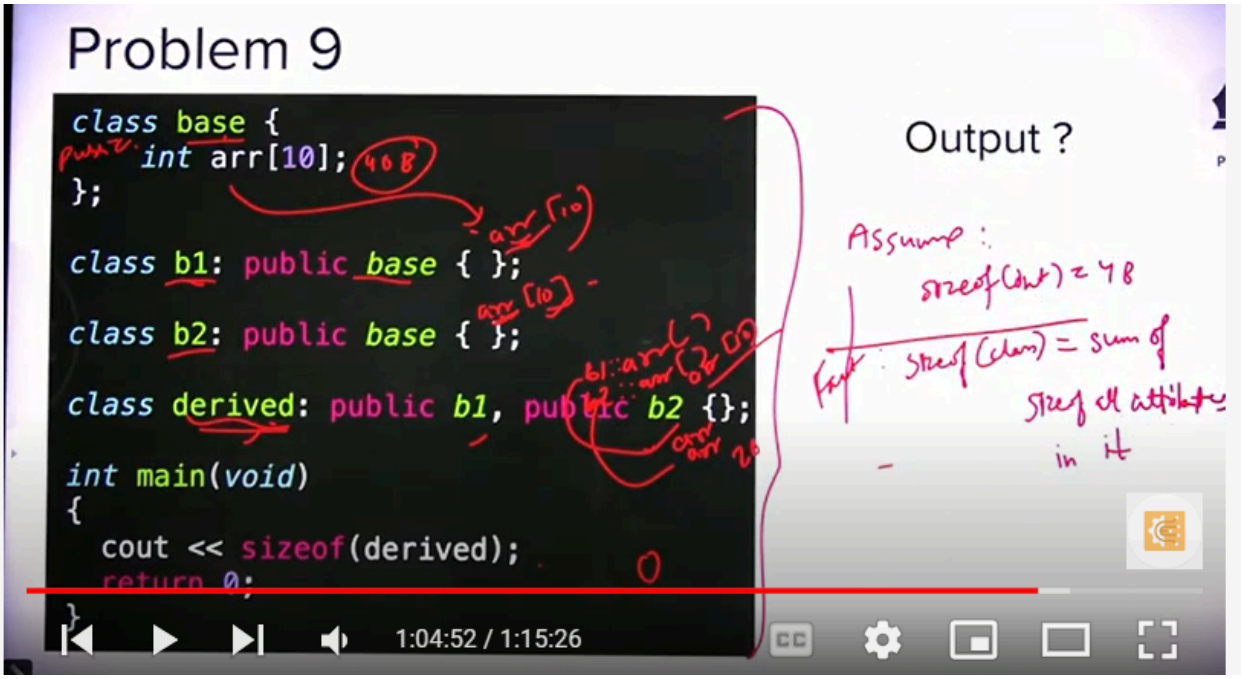
	can be accessed outside of the clas	can be inherited
public:	yes	yes
protected	No	yes
private	NO	No

Q8:



Destructor are called in the reverse order in which constructor are called.

Q9 :



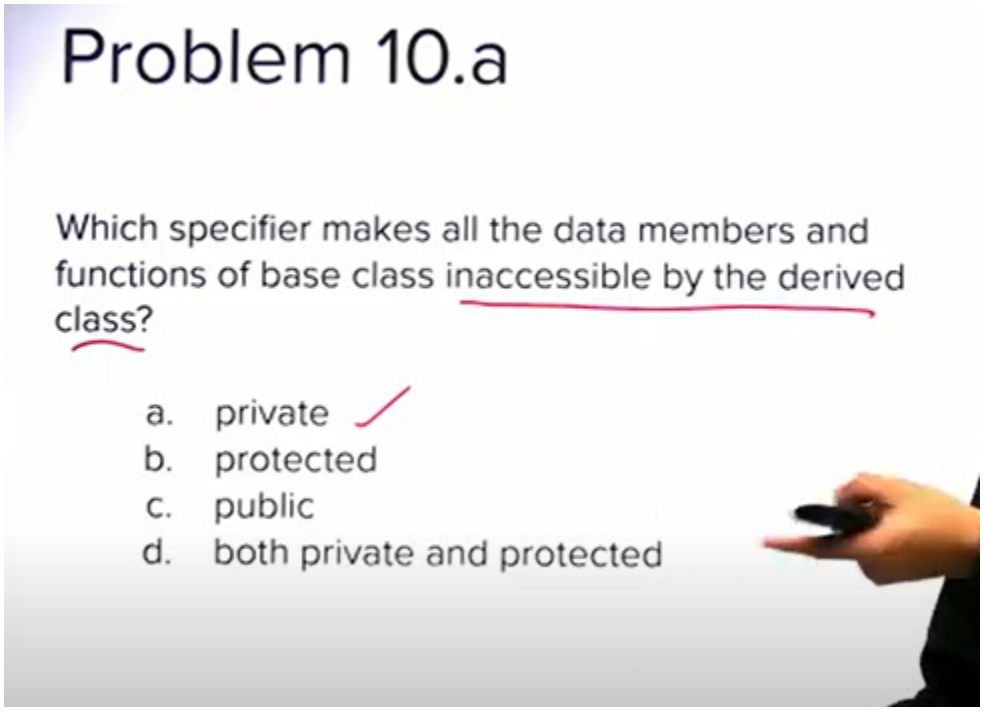
2 arr will be present in the derived class.
There name will be same, hence we will use the class name to resolve the conflict.

```
 9  #include <iostream>
10  using namespace std;
11  class A
12  {
13      public:
14          int arr[10];
15  };
16  class B1 : public A
17  {
18  };
19  class B2 : public A
20  {
21  };
22  class derived : public B1,public B2
23  {
24  };
25  int main()
26  {
27      cout<<sizeof(derived);
28      return 0;
29  }
30
```

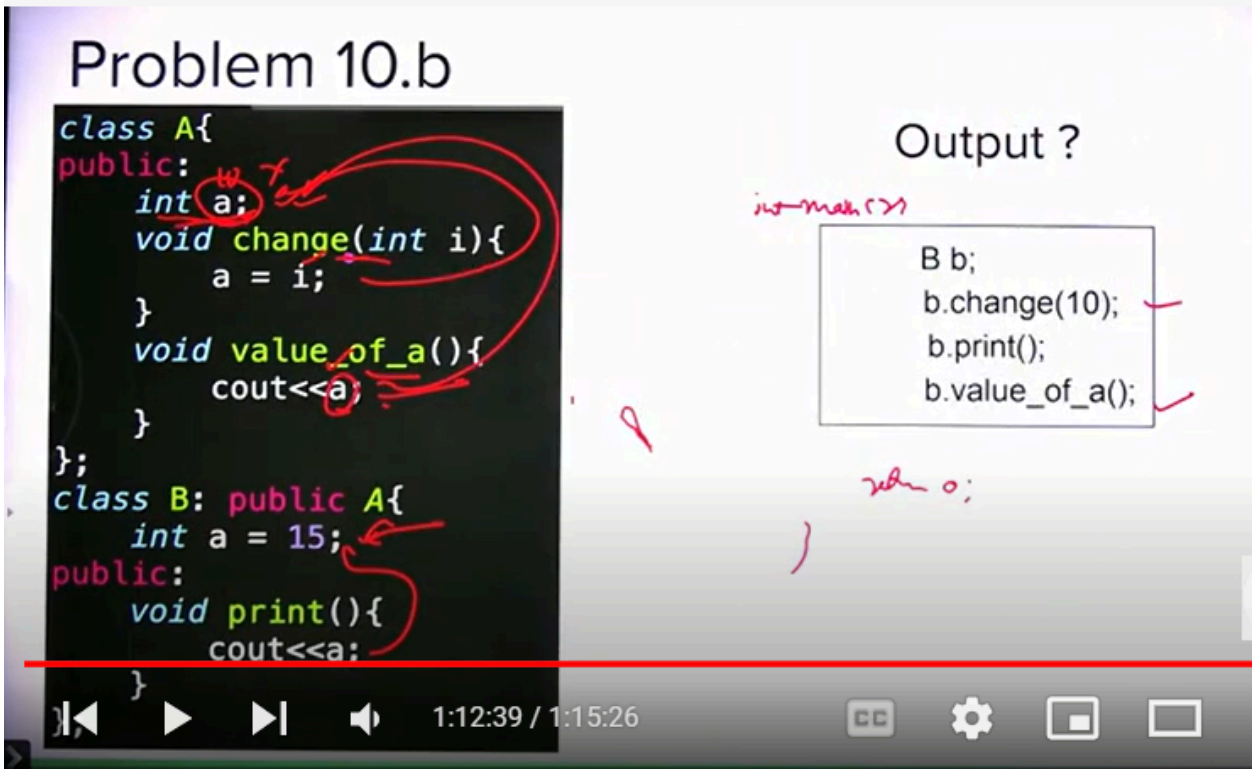
```
80

...Program finished with exit code 0
Press ENTER to exit console.
```

Q10 :



.....



Op: 15 10

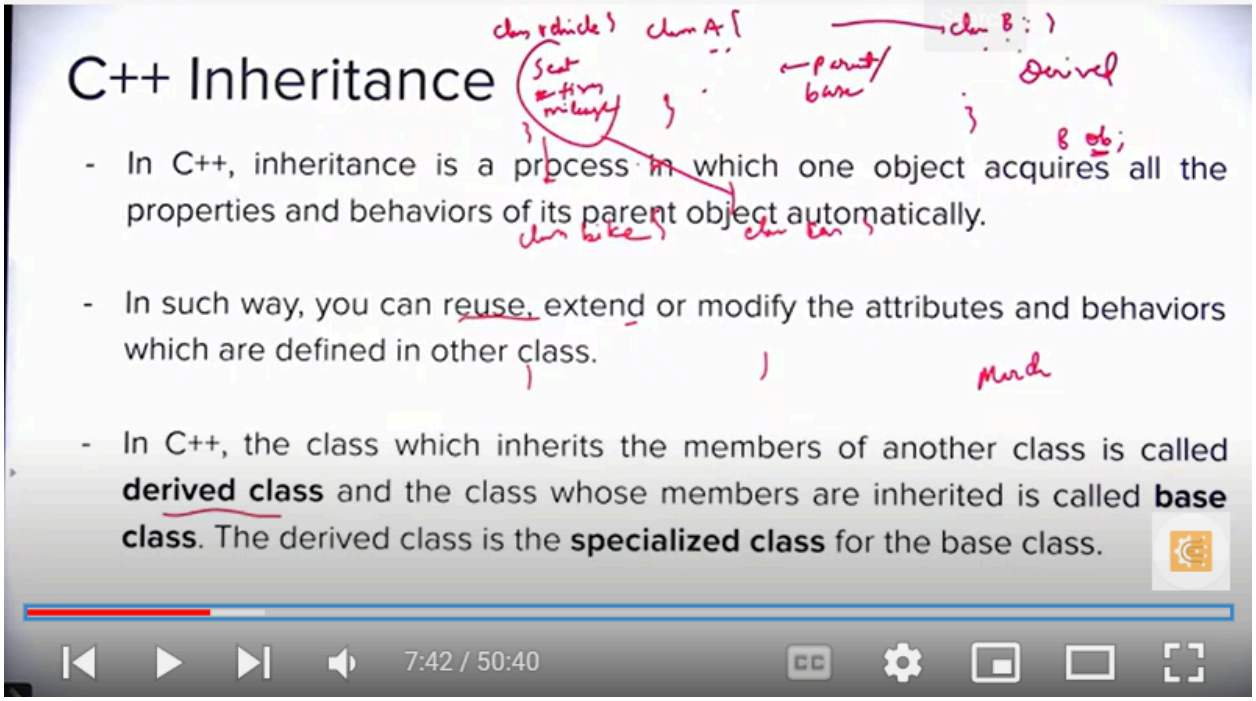
Note: the base class method which is inherited,will access the base class variable only.

So when b.change() will be called, variable a which are present in the class A will be accessed by the change method.
b.print()--> it will access the variable a which are present in the class B
b.value_of_a()---> it will access the variable in class A.

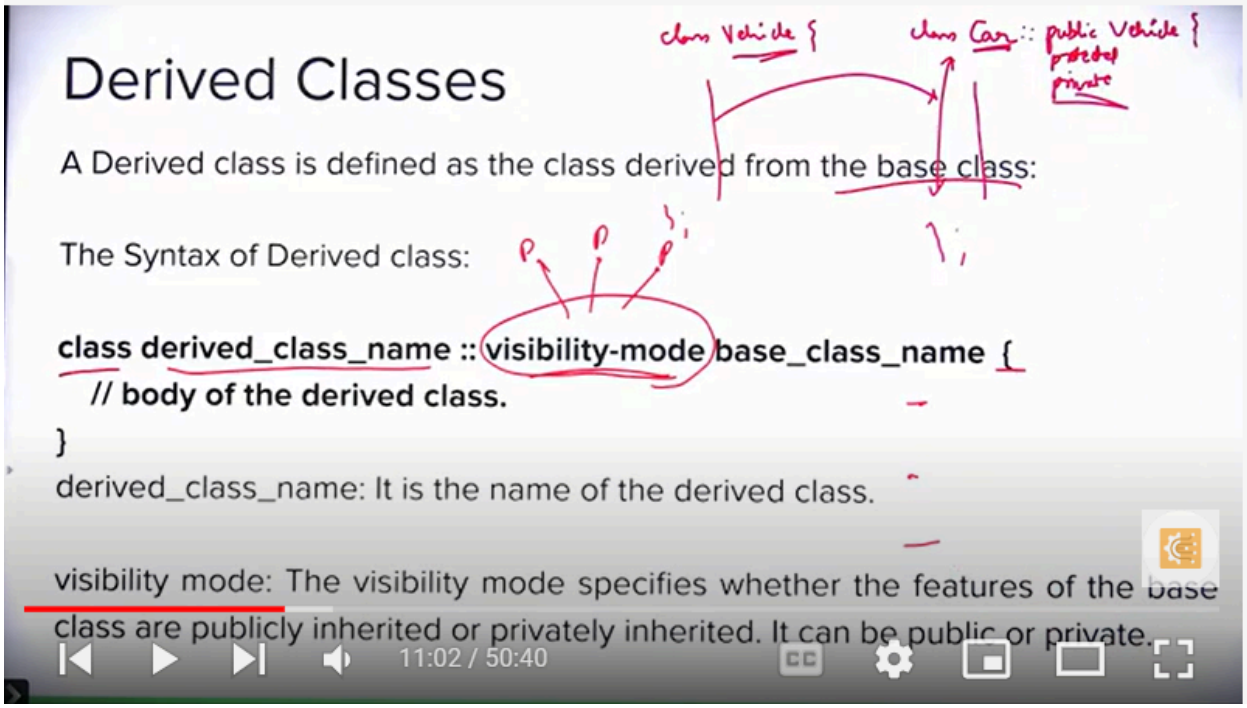
==========
==========

Inheritance | L:21 | C++ | Competitive Programming | Ravindrababu Ravula | Jay Bansal

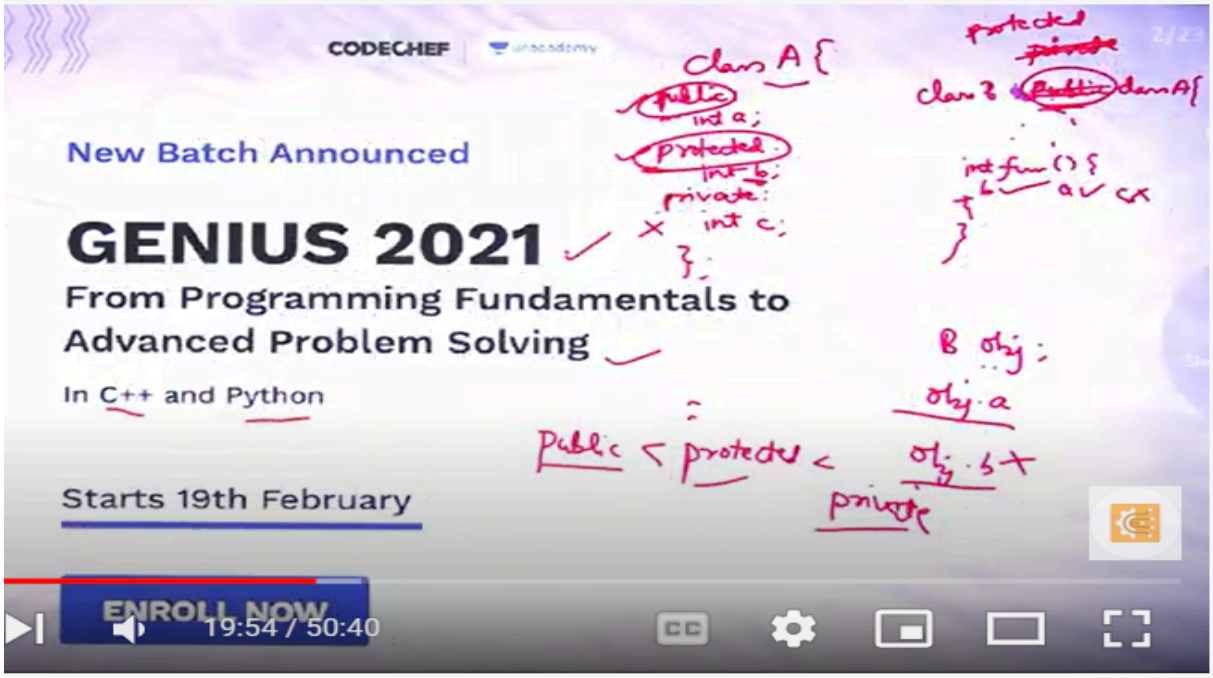
—----------



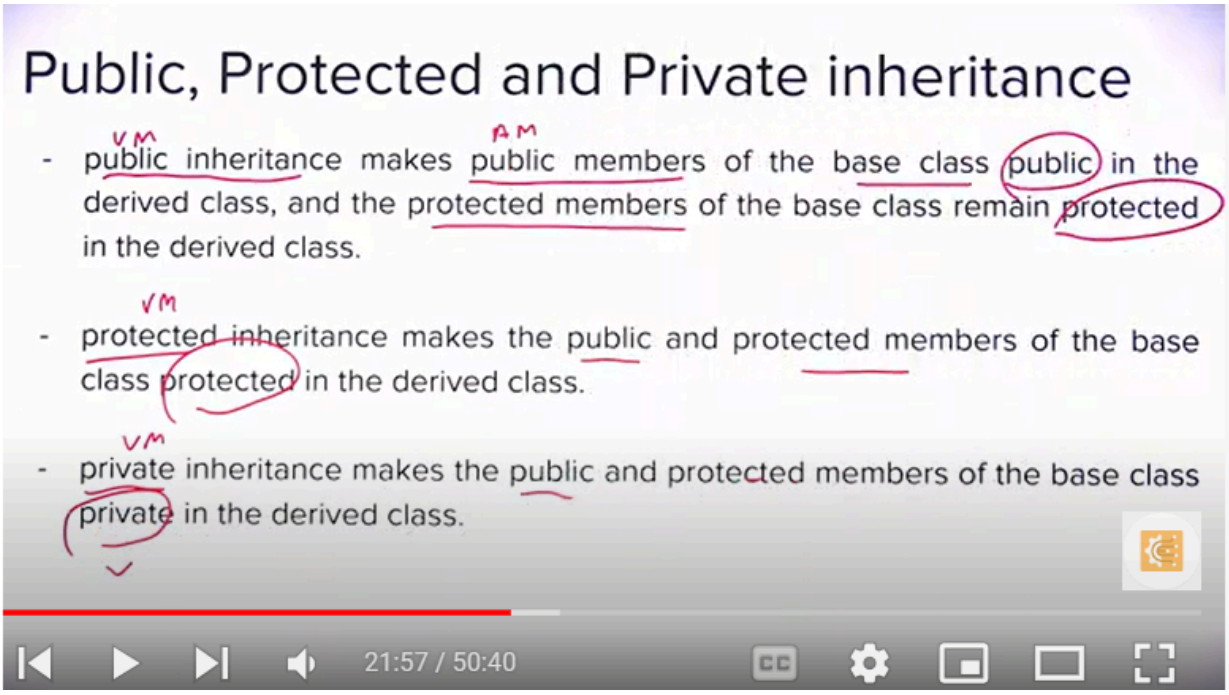
……..



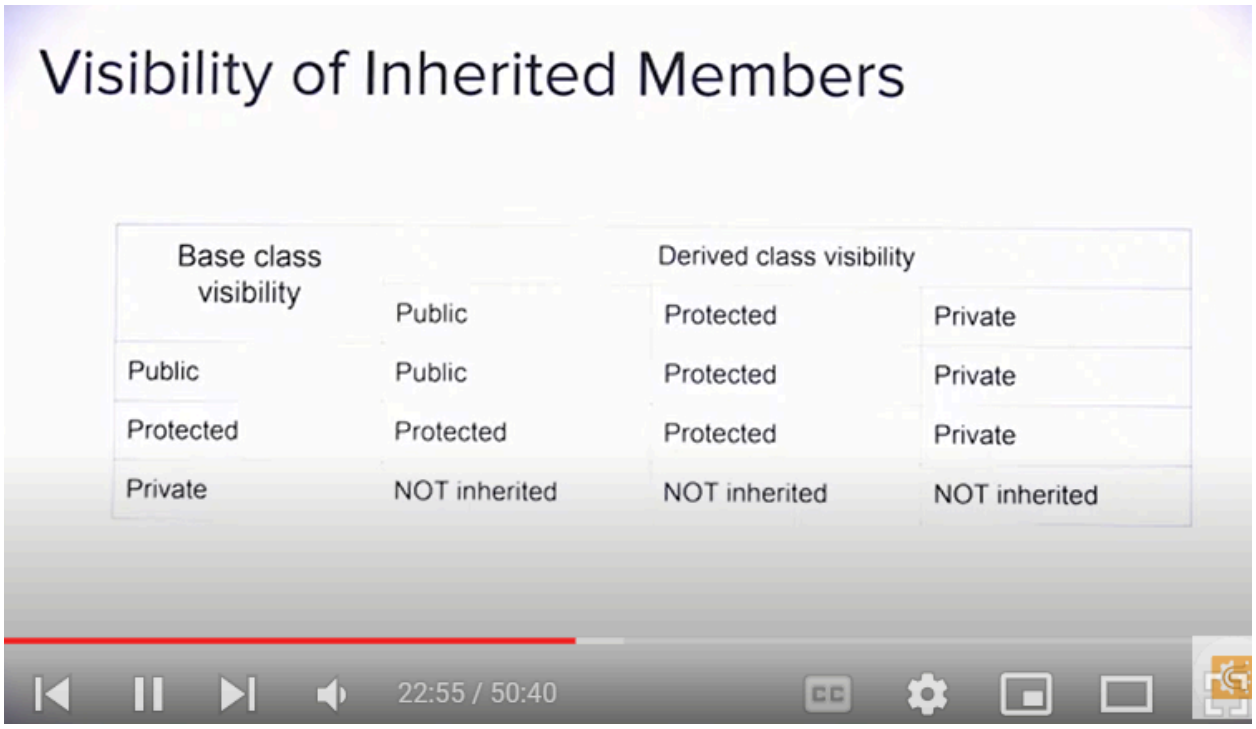
.....



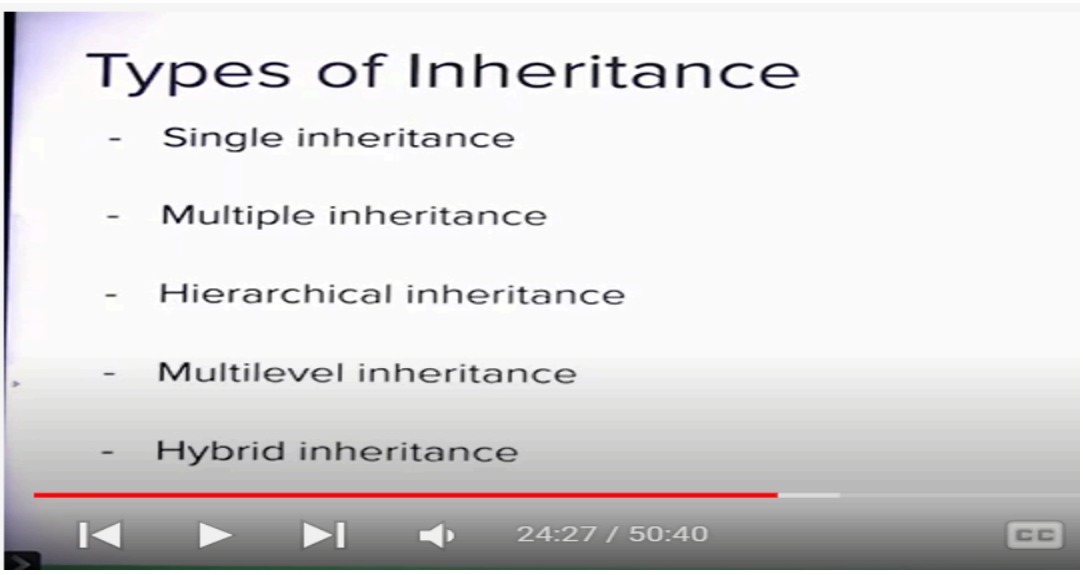
We have to take the maximum security.


Public, Protected and Private inheritance
- public inheritance makes public members of the base class public in the derived class, and the protected members of the base class remain protected in the derived class.
- protected inheritance makes the public and protected members of the base class protected in the derived class.
- private inheritance makes the public and protected members of the base class private in the derived class.
21:57 / 50:40

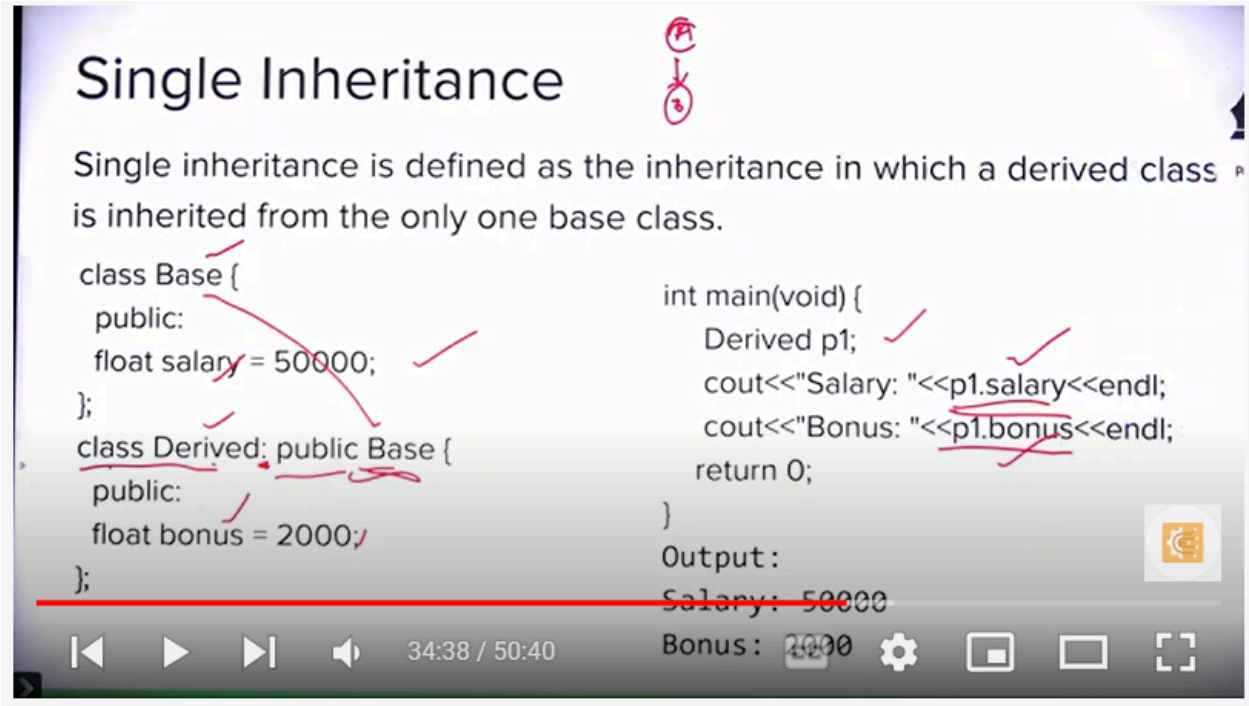
.....

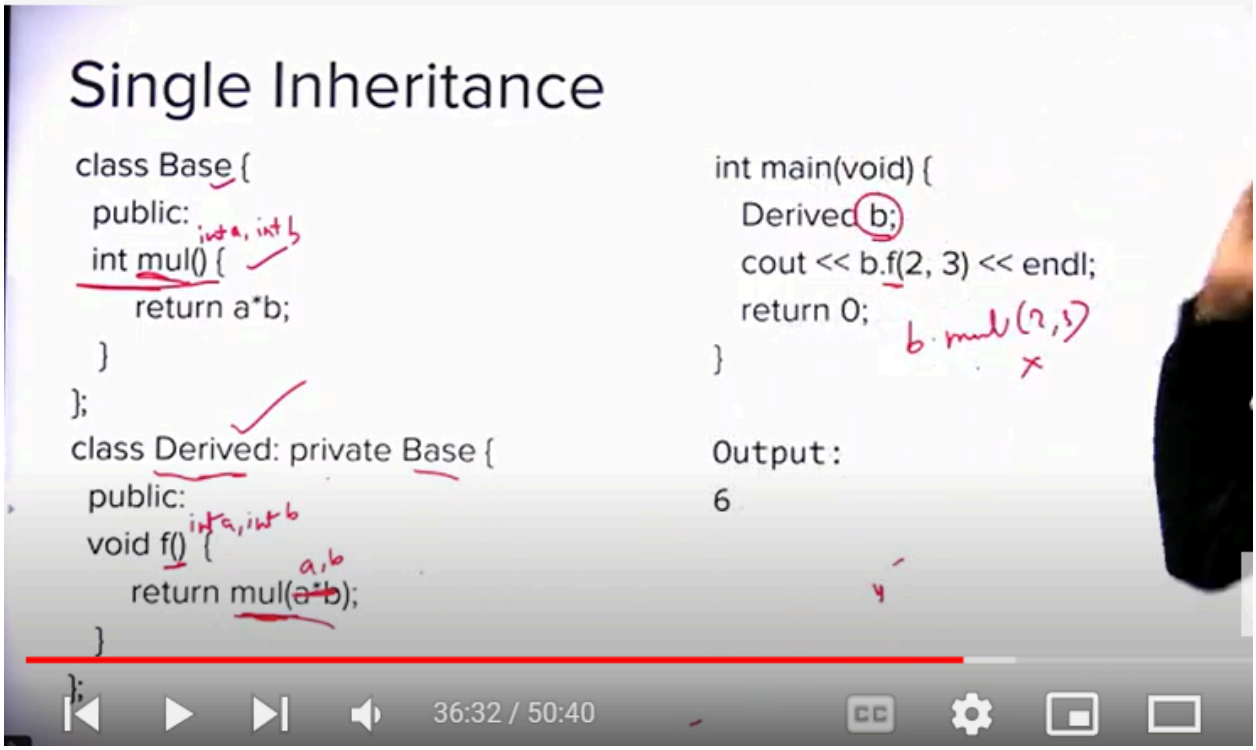

Visibility of Inherited Members
Base class visibility
Derived class visibility
Public
Protected
Private
Public
Public
Protected
Private
Protected
Protected
Protected
Private
Private
NOT inherited
NOT inherited
NOT inherited
22:55 / 50:40

.....

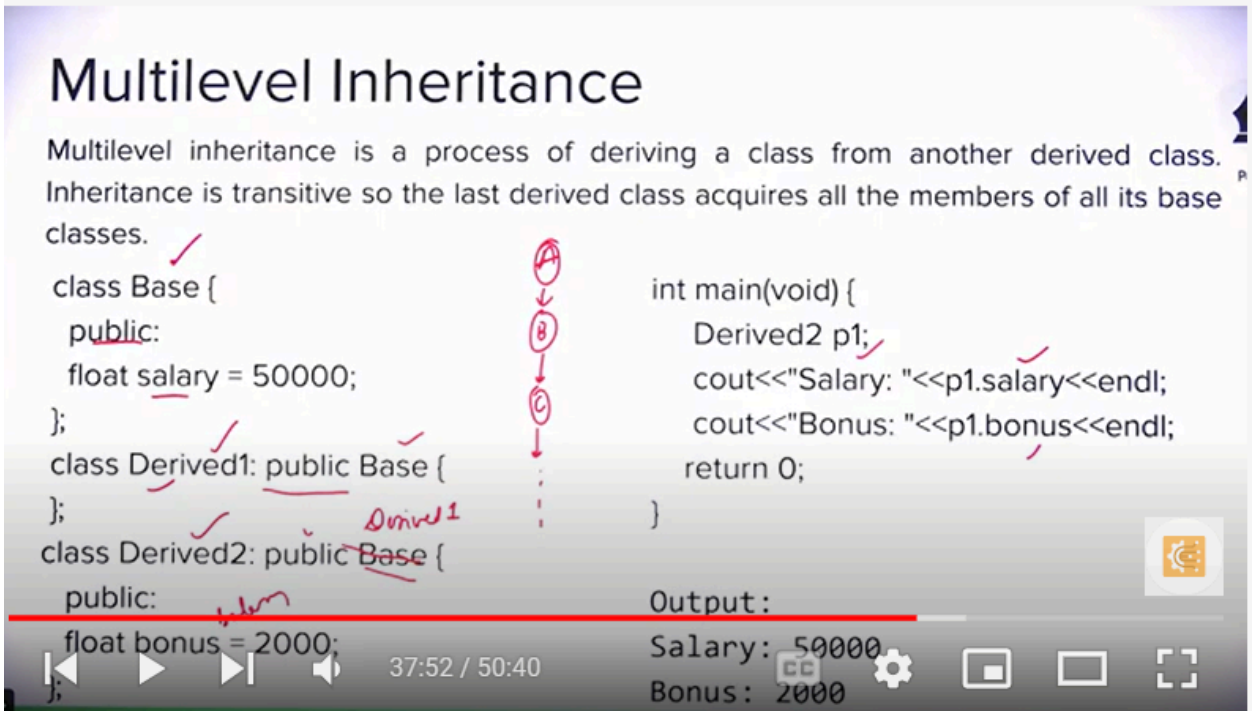

Types of Inheritance
- Single inheritance
- Multiple inheritance
- Hierarchical inheritance
- Multilevel inheritance
- Hybrid inheritance
24:27 / 50:40
CC

/.....

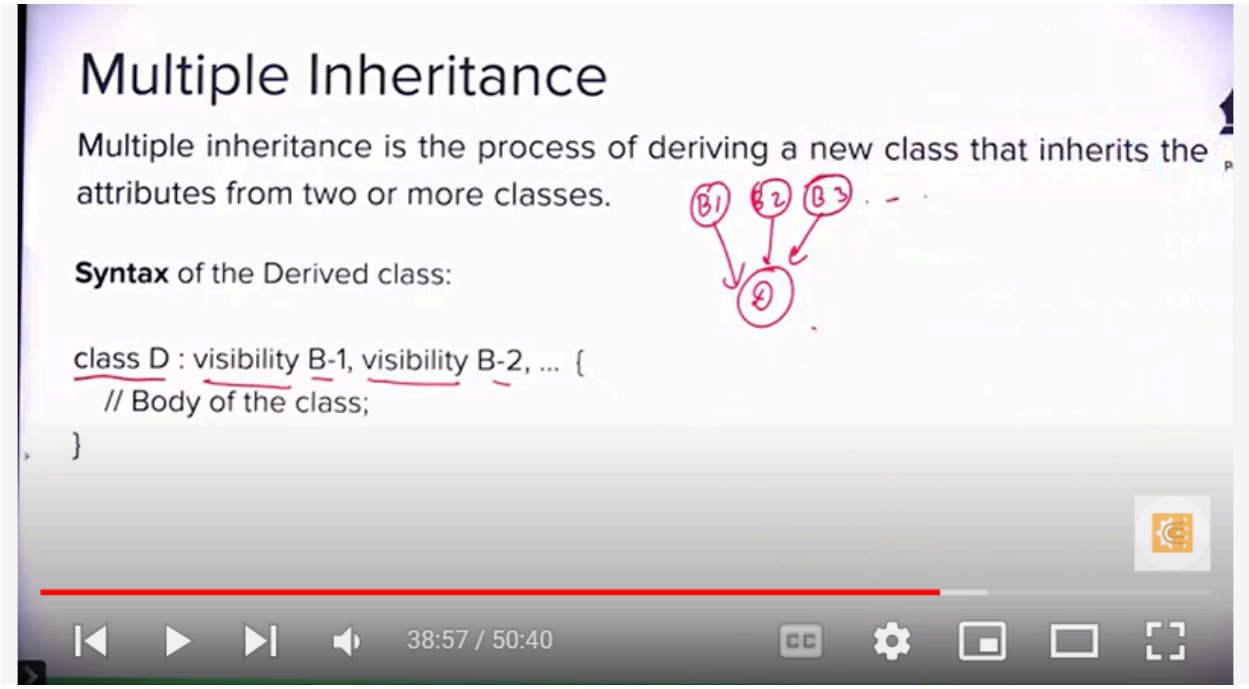

Single Inheritance
Single inheritance is defined as the inheritance in which a derived class is inherited from the only one base class.
class Base {
public:
float salary = 50000;
};
class Derived: public Base {
public:
float bonus = 2000;
};
int main(void) {
Derived p1;
cout<<"Salary: "<<p1.salary<<endl;
cout<<"Bonus: "<<p1.bonus<<endl;
return 0;
}
Output:
Salary: 50000
Bonus: 2000
34:38 / 50:40


Single Inheritance
class Base {
public:
int mul() {
return a*b;
}
};
class Derived: private Base {
public:
void f() {
return mul(a*b);
}
};
int main(void) {
Derived b;
cout << b.f(2, 3) << endl;
return 0;
}
Output:
6
36:32 / 50:40

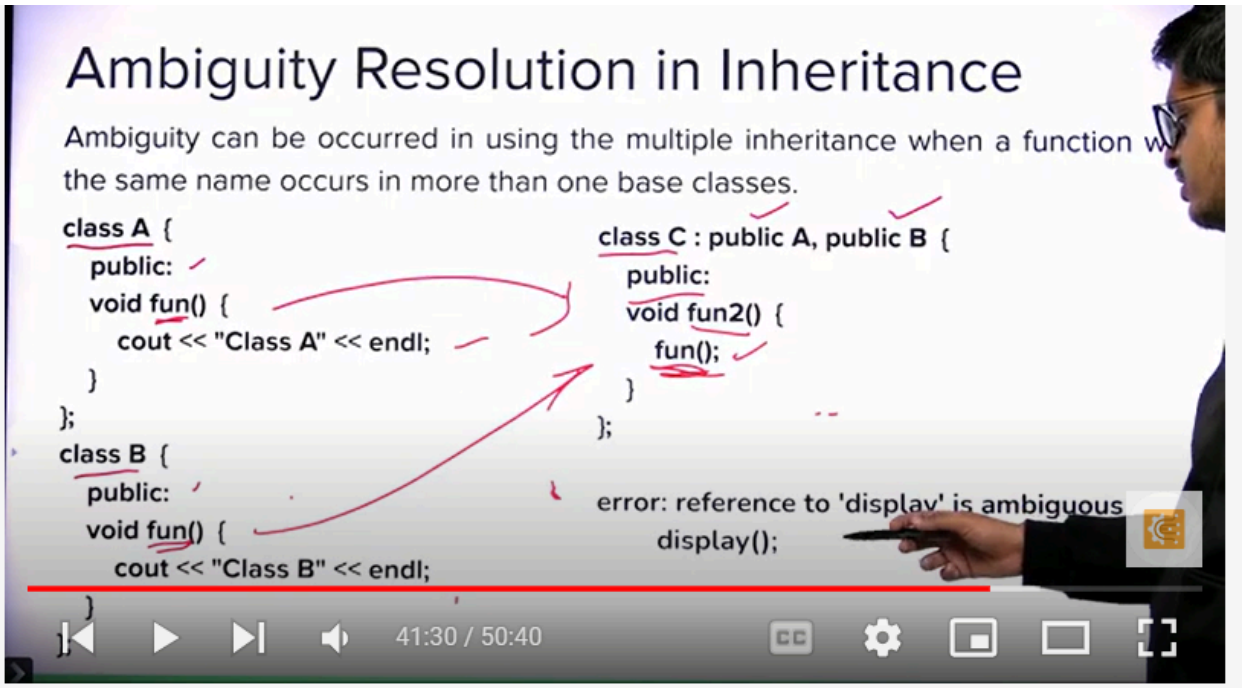
.....


Multilevel Inheritance
Multilevel inheritance is a process of deriving a class from another derived class. Inheritance is transitive so the last derived class acquires all the members of all its base classes.
class Base {
public:
float salary = 50000;
};
class Derived1: public Base {
};
class Derived2: public Base {
public:
float bonus = 2000;
};
int main(void) {
Derived2 p1;
cout<<"Salary: "<<p1.salary<<endl;
cout<<"Bonus: "<<p1.bonus<<endl;
return 0;
}
Output:
Salary: 50000
Bonus: 2000
37:52 / 50:40

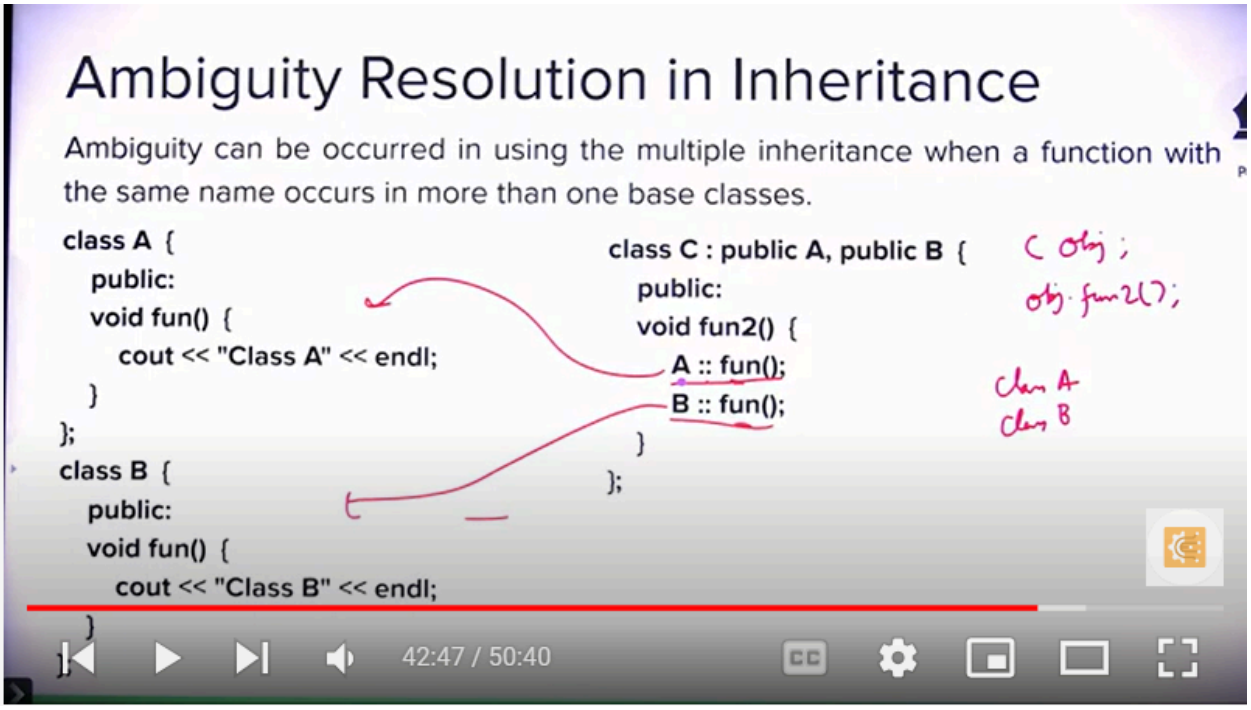
.....



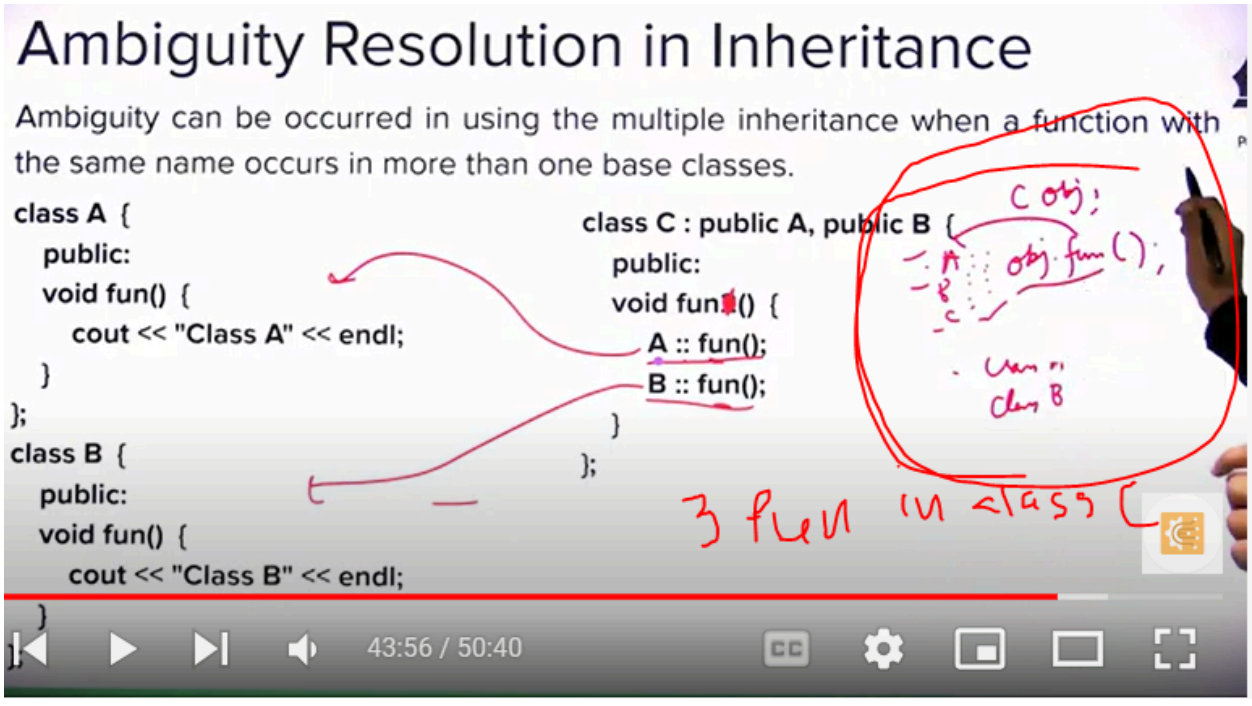
.....



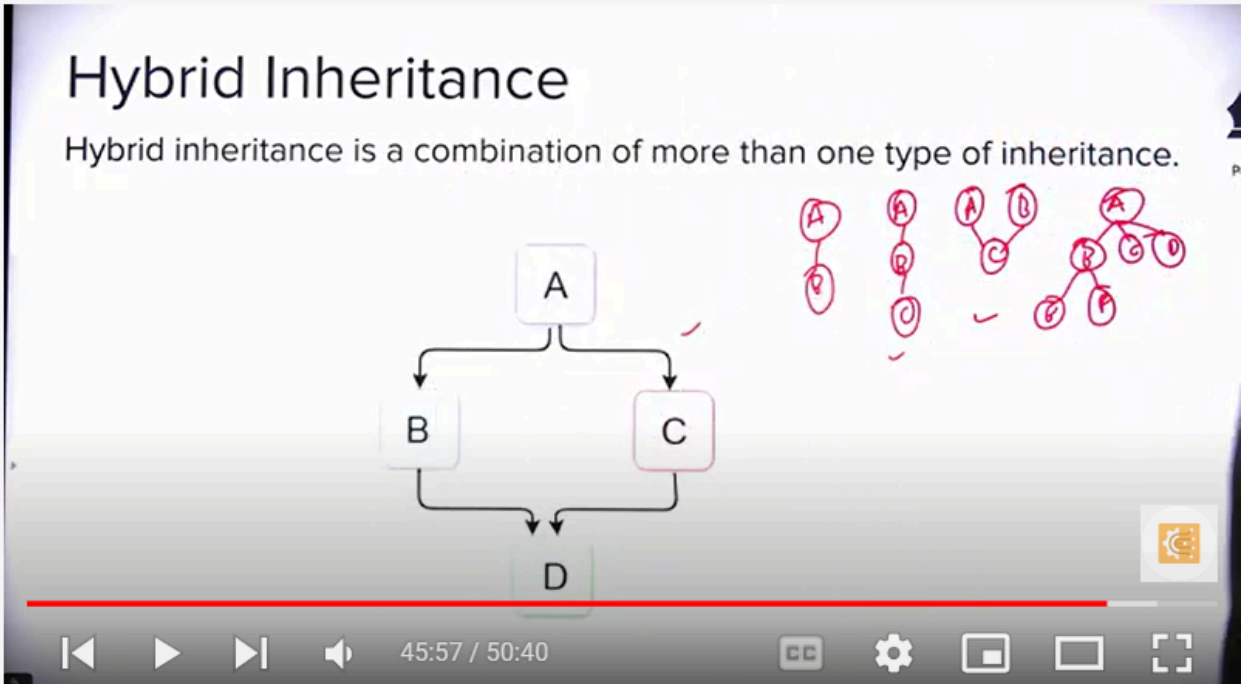
In class C there are 2 fun() which are derived from the class A and B, so it will be ambiguous.
So we have to resolve the ambiguous by using the class name.


Ambiguity Resolution in Inheritance
Ambiguity can be occurred in using the multiple inheritance when a function with the same name occurs in more than one base classes.
class A {
public:
void fun() {
cout << "Class A" << endl;
}
};
class B {
public:
void fun() {
cout << "Class B" << endl;
}
};
class C : public A, public B {
public:
void fun2() {
A :: fun();
B :: fun();
}
};
C obj;
obj.fun2();
Class A
Class B
42:47 / 50:40

.....


Ambiguity Resolution in Inheritance
Ambiguity can be occurred in using the multiple inheritance when a function with the same name occurs in more than one base classes.
class A {
public:
void fun() {
cout << "Class A" << endl;
}
};
class B {
public:
void fun() {
cout << "Class B" << endl;
}
};
class C : public A, public B {
public:
void fun() {
A :: fun();
B :: fun();
}
};
C obj;
obj.fun();
Class A
Class B
3 fun in class C
43:56 / 50:40

.....


Hybrid Inheritance
Hybrid inheritance is a combination of more than one type of inheritance.
A
B
C
D
45:57 / 50:40

.....