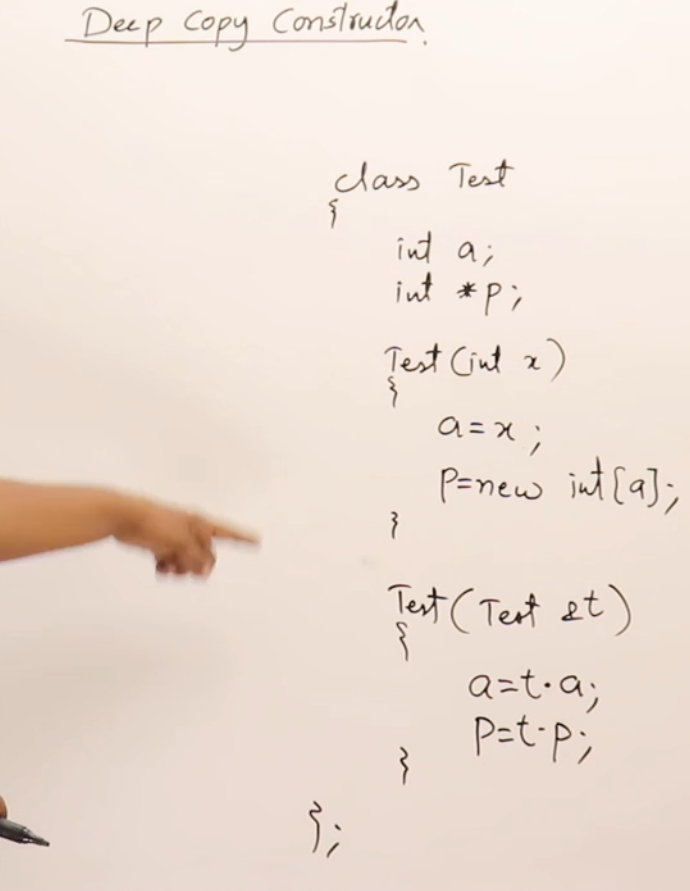
**OOPS**

1. Use of Inheritence – Reusability
2. Use of Overriding (redefining functions in derived class) – To achieve Polymorphism
3. Abstract class - containing atleast one pure virtual function – it’s object can’t be created but it’s pointer can be created !!!
4. Derived class of abstract class has to override pure virtual functions else it will also become an abstract class.
5. **Constructors in Inheritence** : If default constructor of derived class is called, then default constructor of base class is executed first and then default constructor of derived class is executed
6. If parameterized constructor of derived class is called, then first default constructor of base class is executed and then parameterized constructor of derived class
7. If you want to call parameterized constructor of base class then in parameterized constructor of derived class, write it as

( : public base)

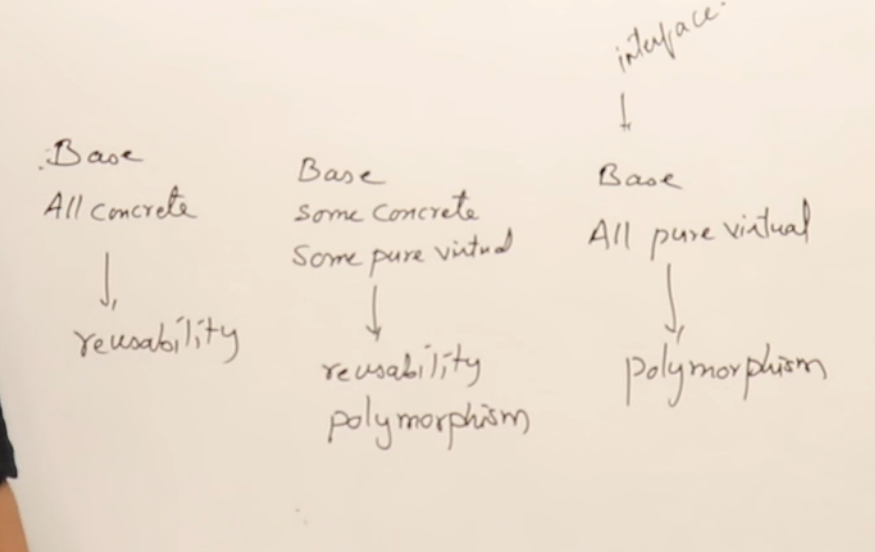
1. Copy constructor – (Rectagle(&Rectangle rec){ l = rec.l;}



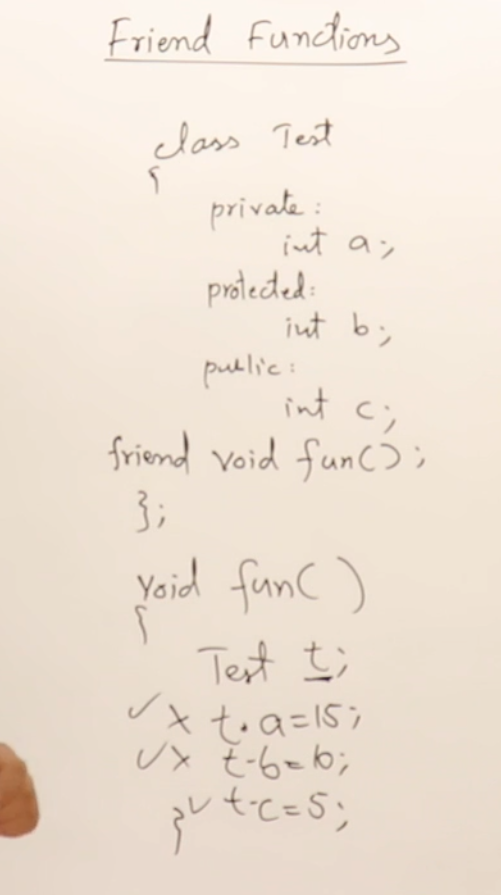
1. In base class pointer pointing to derived class object, only the functions of base class can be called, and if we class any overriding function, then the base class function will get executed.
2. But if you want derived class’ overriding function to get executed, use “virtual” keyword in front of base class’ overrided function !!!! -- (Polymorphism)
3. Derived class pointer pointing to base class object cannot be created !!
4. Inline function – whose function definition expands in the same line where it is called.

A function whose definition is written inside the class only – inline

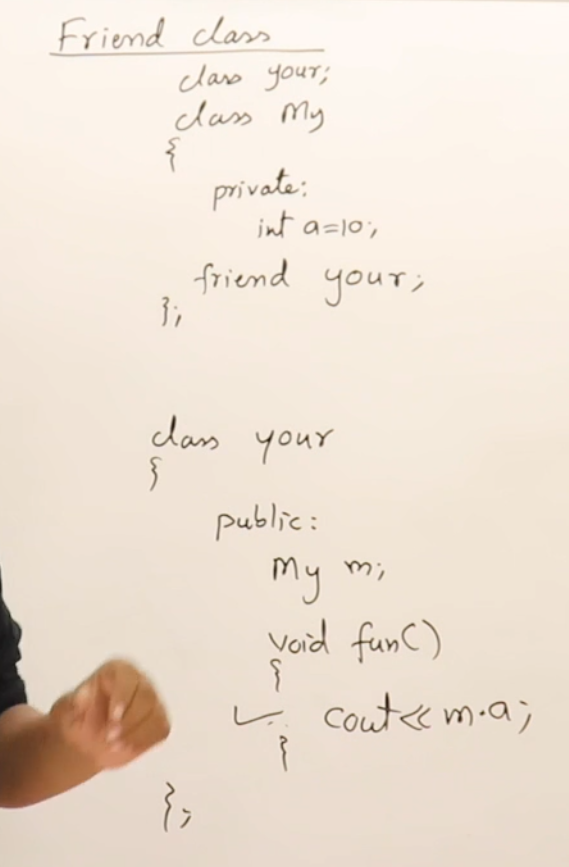
1. A function which is declared inside the class but defined outside using scope resolution operator – non inline and to make it inline – use inline keyword in front of where it is declared inside the class.
2. Three types of classes :



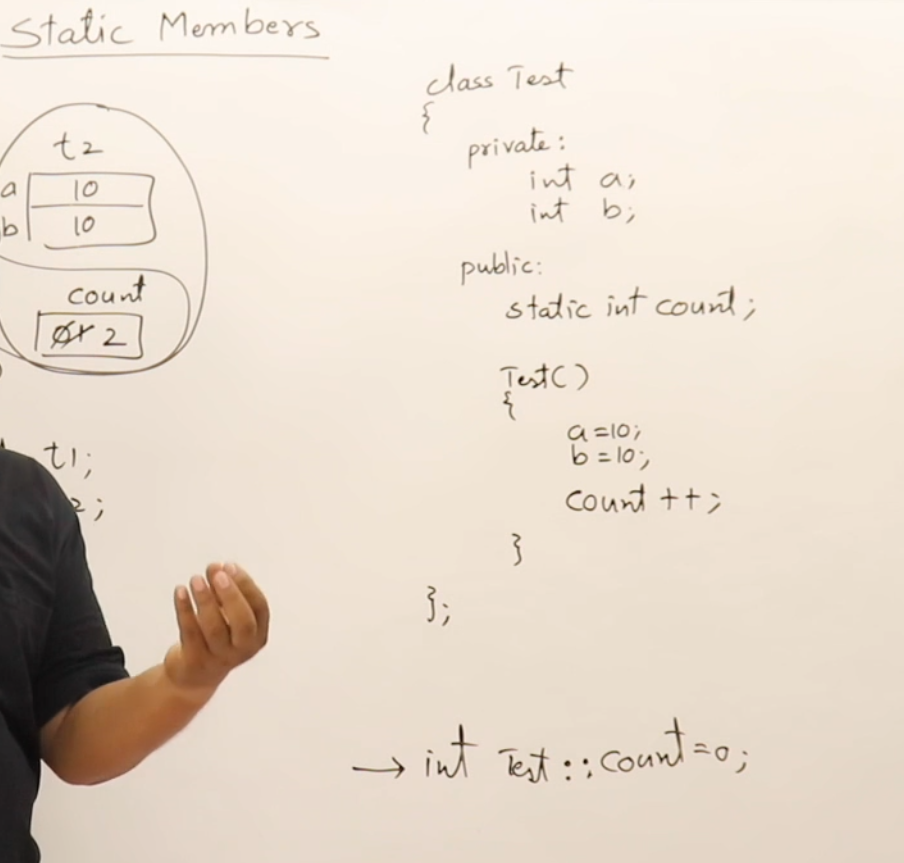
1. In C, we cannot write functions inside a structure but in C++, we can even write functions inside a structure
2. Difference between struct and class in C++ is that in struct, all members by default are public while in class, they are private.
3. **Friend Function :**



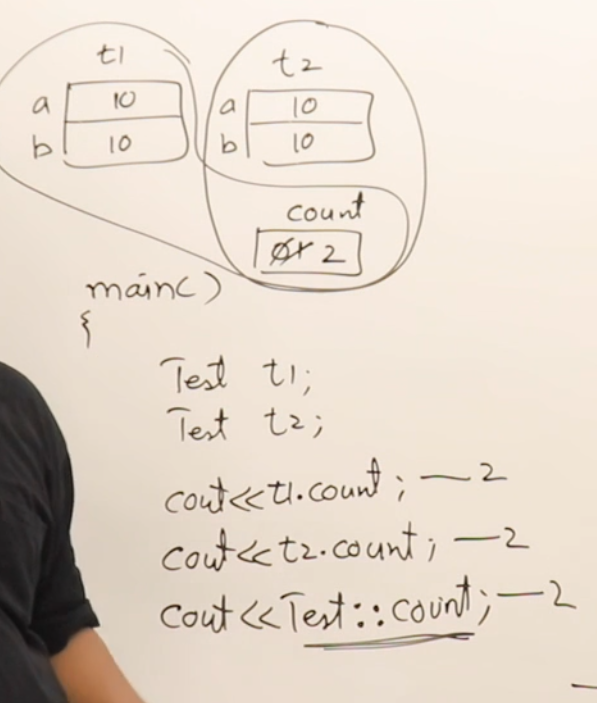
1. **Friend Class :**

****

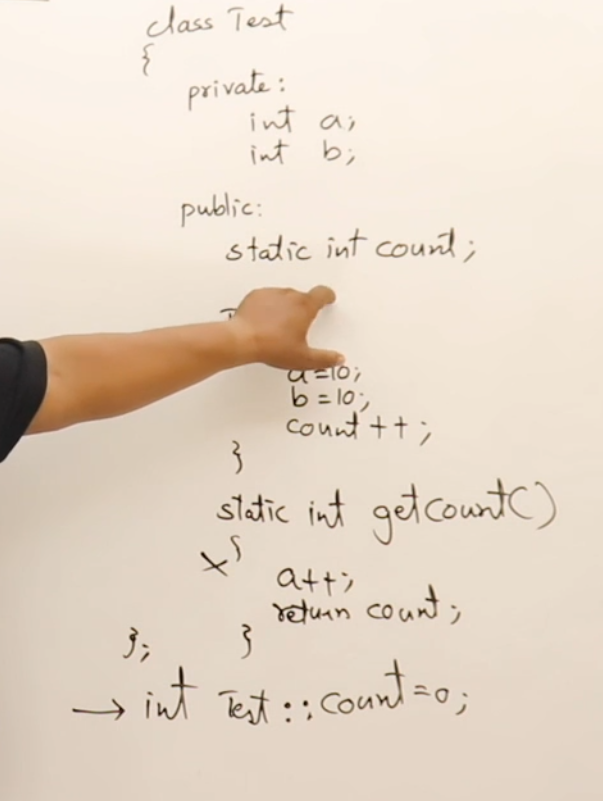
1. An Object can access only public members of a class and an inheriting class can access protected as well as public members of the parent class !!!!
2. Static data members **: they must be declared two times, one time inside the class and second time outside the class using scope resolution operator**

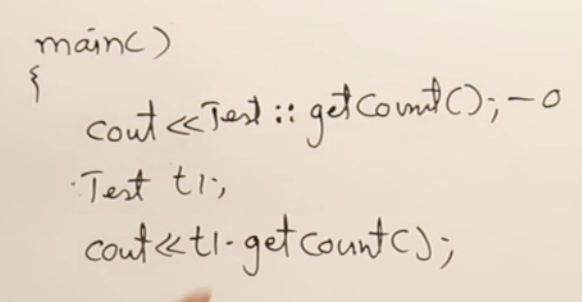
****

1. Static data members can be accessed using object as well as using class name using scope resolution operator

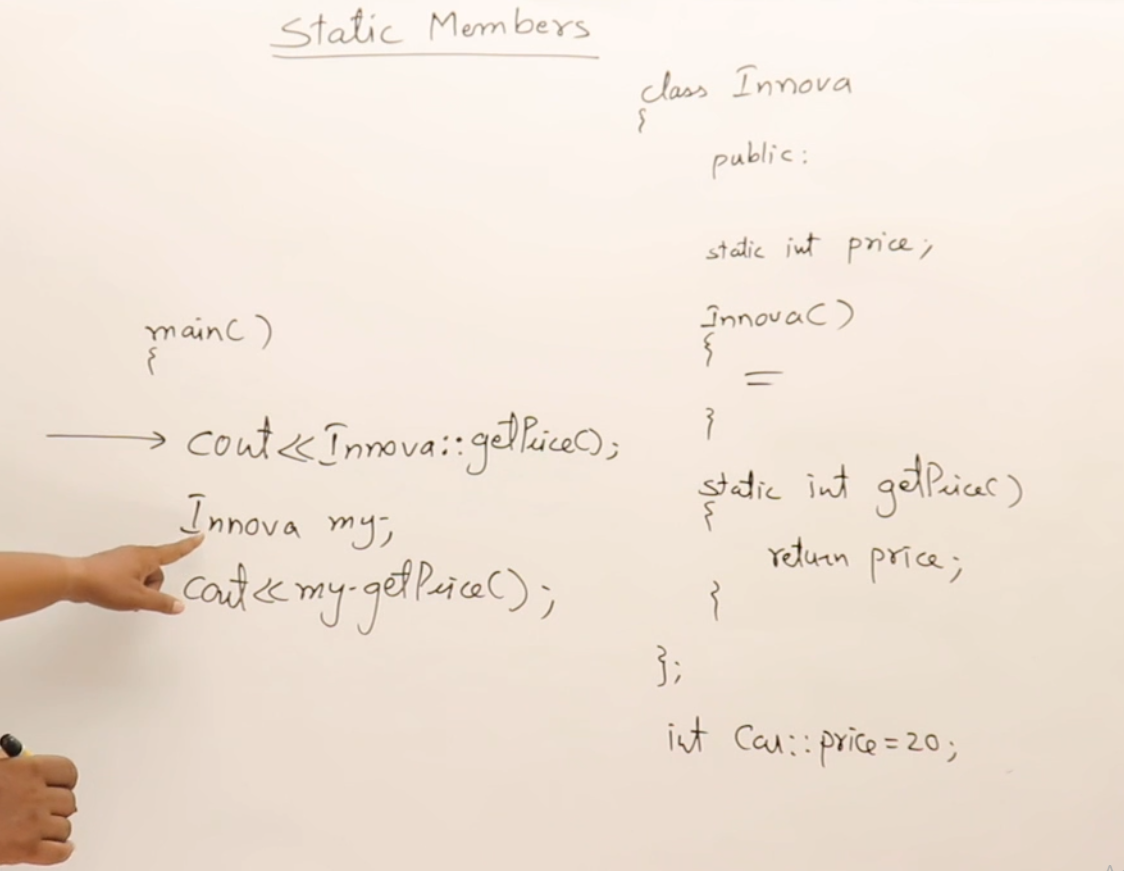


1. **Static member functions :**  they can only access static data members and not non static data members !! These can also be called using class name with scope resolution operator

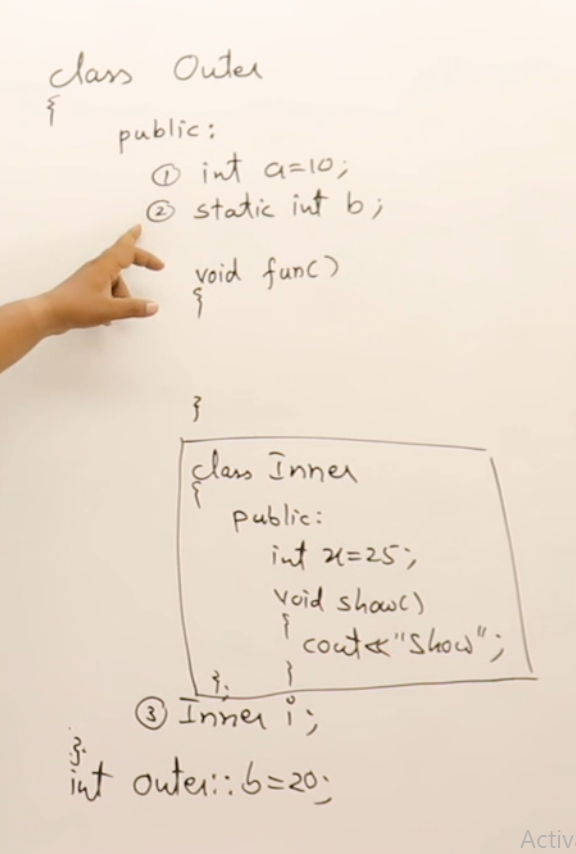
****

****

1. **Non static member functions can also access static data members but static member functions can only access static data members !!!**

****

1. **Inner/ Nested Class :**

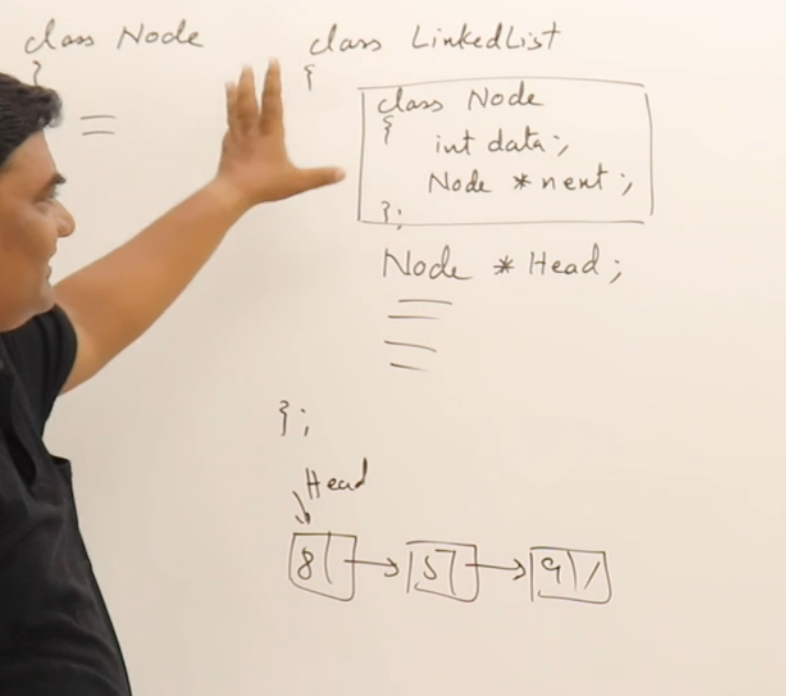
****

**Inner class can only and only access the static data members of Outer class !!!**

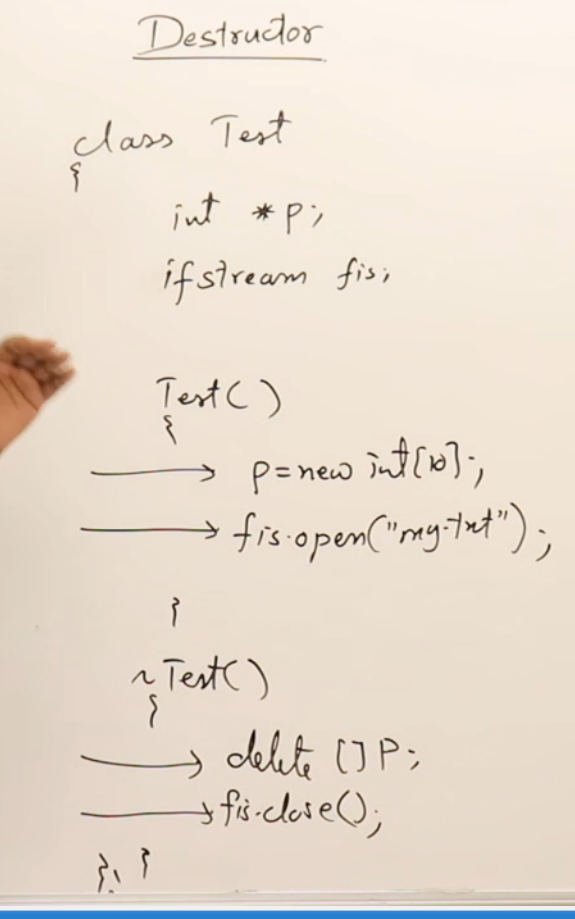
**Outer class can have object of inner class as its member.**

**Outer class can access only public members of the inner class- - (has a relationship)**

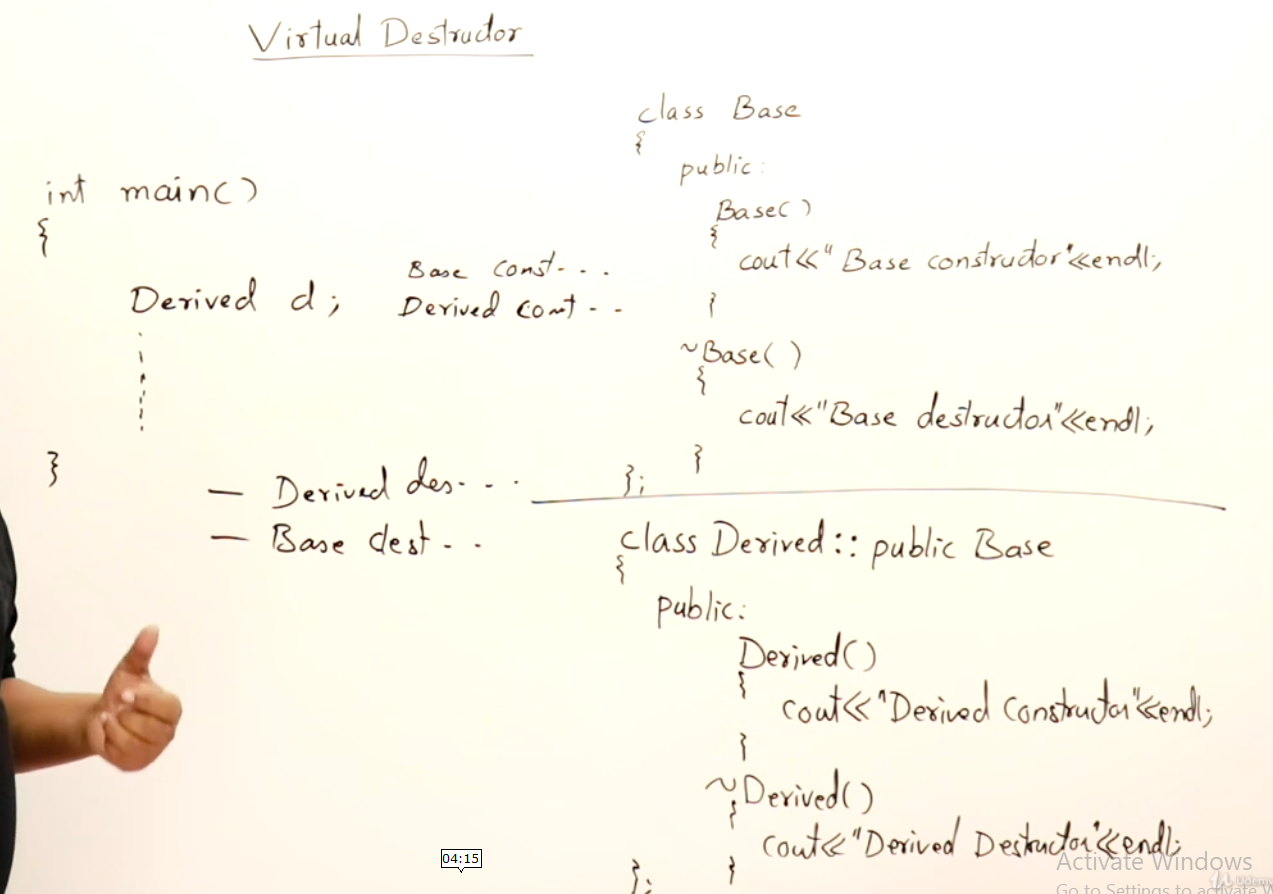
1. Use of inner class :

****

1. Destructors :

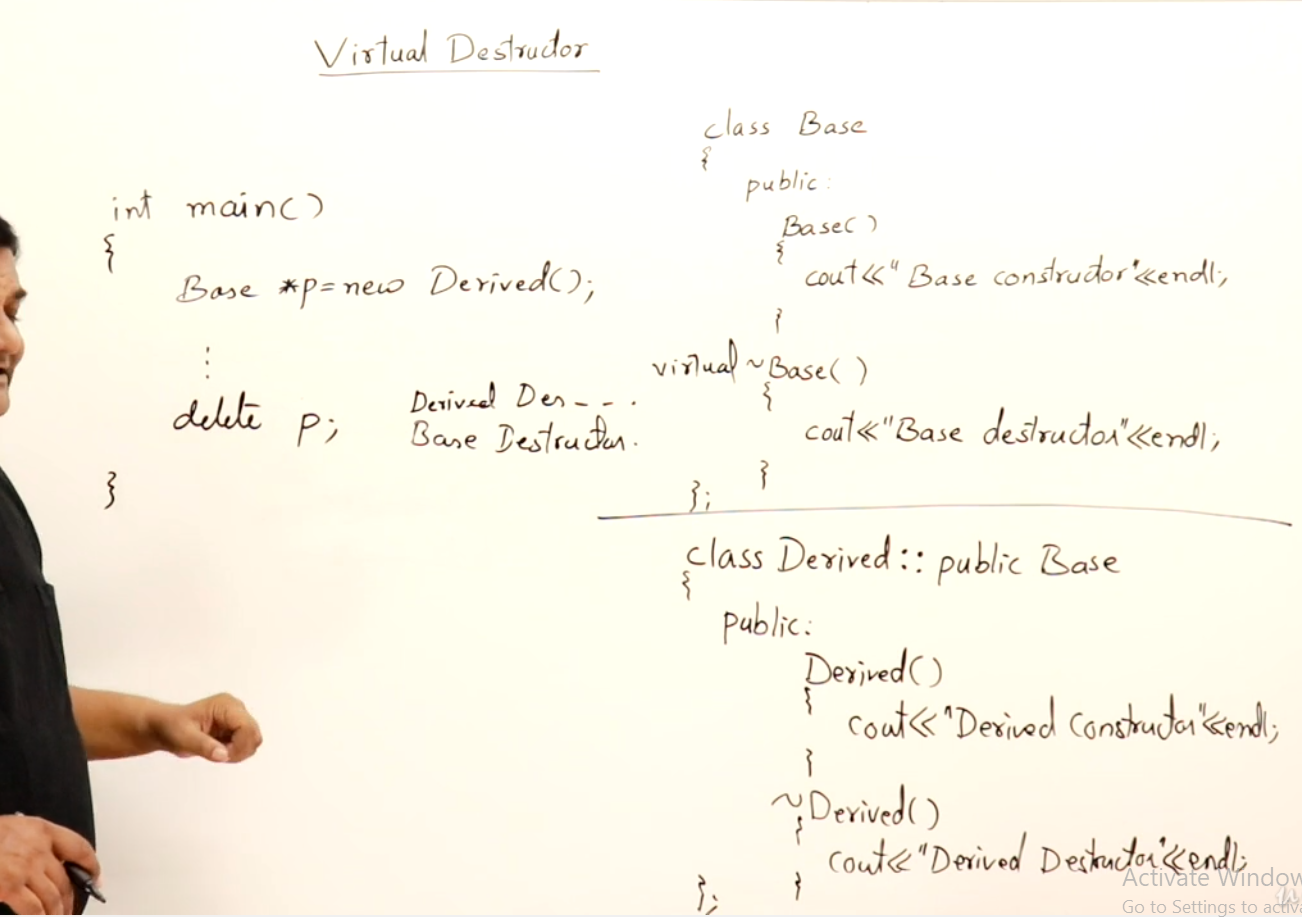
****

1. We can overload the constructor and can have multiple constructors but we cannot have multiple destructors. Both constructor and destructor don’t return anything and both have same name as their class . Destructor cannot be overloaded and cannot have any argument !!!! A destructor can be virtual also. Derived class destructor is executed first and then base class unlike constructors.

****

1. If pointer is of base class pointing to derived class object, then only base class destructor will be called because in C++, reference to pointer decides the decisions.

But if we want to call derived class destructor also followed by base class destructor, then use “virtual” keyword in front of base class destructor when your pointer is of base class : **(Runtime Polymorphism)**

****