Making a Private Member Inheritance

- We have seen how to increase the capabilities of an existing class without modifying it
- Also we have seen that a private member of a base class can not be inherited and therefore it is not available for the derived class directly.
- What do we do if the private data needs to be inherited by a derived class?
- This can be accomplished by modifying the visibility limit of of the private member by making it public.
- This would make it accessible to all the other functions of the program, this eliminate the advantage of data hiding.

Making a Private Member Inheritance

- C++ provides a third visibility modifier,protected, which serve a limited purpose inheritance.
- A member declared as protected is accessible by the member functions within its class and any class immediately derived from it.
- It can not be accessible by the functions outside these two classes.

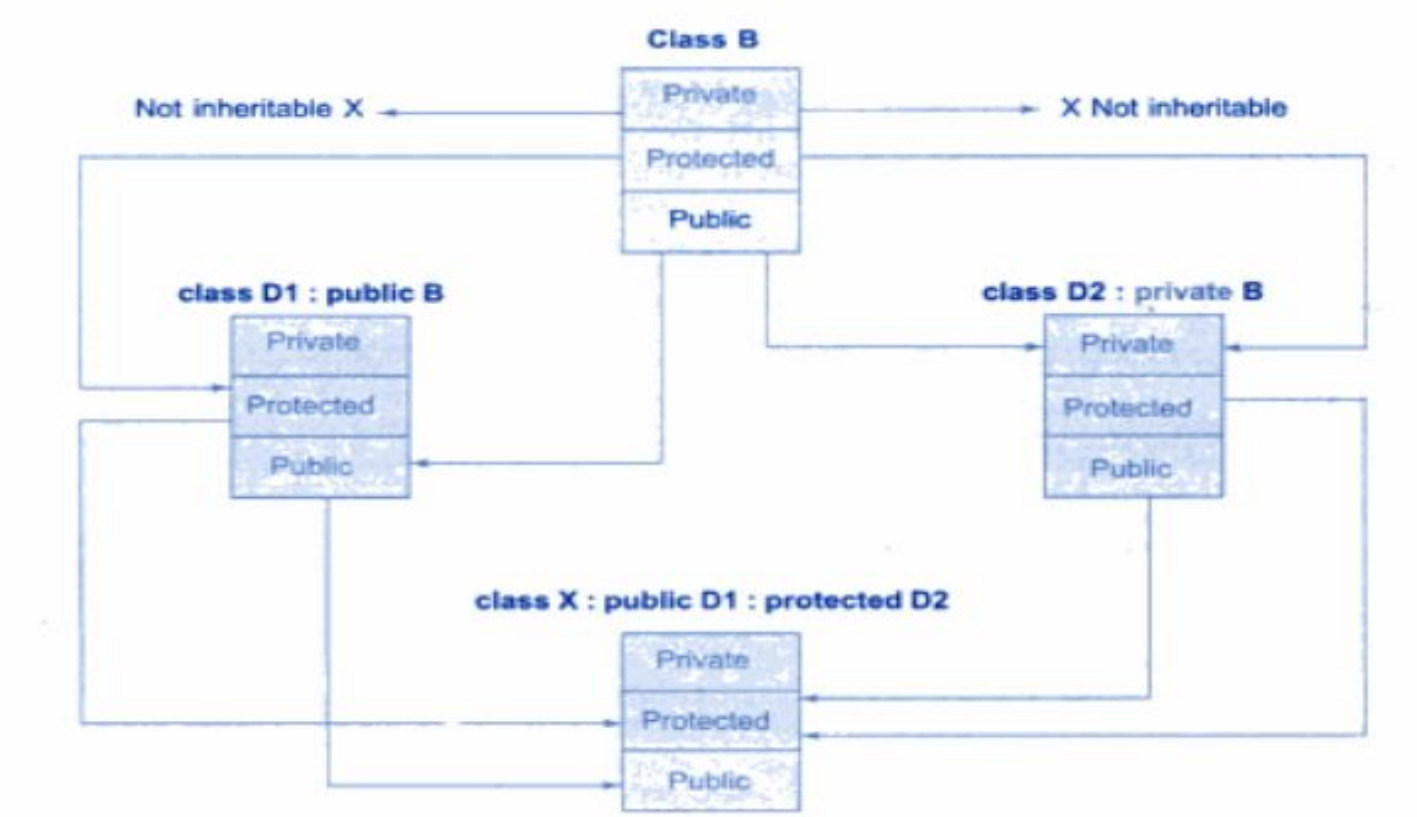
```
class sample
{
   private:
      . . . .
      . . . .
   protected:
      . . . .
      . . . .
   public:
      . . . .
      . . . .
}
```

Making a Private Member Inheritance

Base class visibility	*Derived class visibility*		
	Public derivation	***Private derivation***	***Protected derivation***
Private ⟶	Not inherited	Not inherited	Not inherited
Protected ⟶	Protected	Private	Protected
Public ⟶	Public	Private	Protected

- When a protected member is inherited in public mode,it becomes protected in derived class too.
- When a protected member is inherited in private mode, it becomes private in derived class.

Making a Private Member Inheritance



Multilevel Inheritance

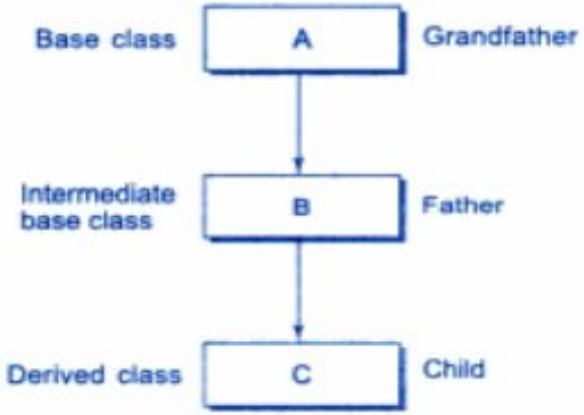


Fig. 8.7 ⇔ *Multilevel inheritance*

A derived class with multilevel inheritance is declared as follows:

```
class A{.....};                 // Base class
class B: public A {.....};      // B derived from A
class C: public B {.....};      // C derived from B
```

This process can be extended to any number of levels.

Multilevel Inheritance

```
class student
{
      protected:
      int roll_number;
   public:
      void get_number(int);
      void put_number(void);
};
```

```
class test : public student
{
  protected:
      float sub1;
      float sub2;
  public:
      void get_marks(float, float);
      void put_marks(void);
};
```

```
class result : public test
{
      float total;
  public:
      void display(void);
};
```

Multilevel Inheritance

The class result, after inheritance from 'grandfather', through 'father', would contains the following members

```
private:
    float total;                    // own member
protected:
    int roll_number;                // inherited from  student via test
    float sub1;                     // inherited from test
    float sub2;                     // inherited from test
public:
   void get_number(int);                // from student via test
   void put_number(void);               // from student via test
   void get_marks(float, float);        // from test
   void put_marks(void);                // from test
   void display(void);                  // own member
```

Multilevel Inheritance

The inherited functions put_number() and put_mark()can be used in the definition of display() function

```
void result :: display(void)
{
      total = sub1 + sub2;
      put_number();
      put_marks();
      cout << "Total = " << total << "\n";
}
```

Here is a simple **main()** program:

```
int main()
{
      result student1;
      student1.get_number(111);
      student1.get_marks(75.0, 59.5);
      student1.display();

   return 0;
}
```