Public/Protected/Private

Access Level of Inherited Class members

```cpp
using namespace std;
#include <iostream>

class CBox
{
 public :
 void ShowVolume() const
  { cout << endl
         << "CBox usuable volume is "<<Volume();
  }
  // Function to calculate the volume of a CBox object
  //virtual
  double Volume() const
  {
    return m_Length*m_Width*m_Height; }

  //Constructor
  CBox(double lv=1.0, double wv=1.0, double hv=1.0)
                    :m_Length(lv), m_Width(wv), m_Height(hv) {}

   public      :  double m_Length;
   protected   :  double m_Width;
   private     :  double m_Height;

};
```

This example comes from Ivor Horton Visual C++2010 WROX publishers pages 572-577.

```cpp
#include "box.h"

class CGlassBox: public CBox
{
 public :
 void ShowVolume() const
  { cout << endl
         << "CBox usuable volume is "<<Volume();
  }
  // Function to calculate the volume of a CBox object
  //virtual
  double Volume() const
  { return 0.85*m_Length*m_Width*m_Height; }

  //Constructor
  CGlassBox(double lv, double wv, double hv) : CBox(lv, wv, hv){}

  // inherited from box.h as    public      :  double m_Length;

  // inherited from box.h as    protected   :  double m_Width;

  // no access ever             private     :  double m_Height;

};
```

Since this class is ***public*** it inherits the access level the same as the base class.

```cpp
using namespace std;
#include <iostream>

class CBox
{
 public :
 void ShowVolume() const
  { cout << endl
         << "CBox usable volume is "<<Volume();
   }
  // Function to calculate the volume of a CBox object
  //virtual
  double Volume() const
  {
    return m_Length*m_Width*m_Height; }

  //Constructor
  CBox(double lv=1.0, double wv=1.0, double hv=1.0)
                    :m_Length(lv), m_Width(wv), m_Height(hv) {}

   public      :  double m_Length;
   protected   :  double m_Width;
   private     :  double m_Height;

};
```

```cpp
#include "box.h"

class CGlassBox: protected CBox
{
 public :
 void ShowVolume() const
  { cout << endl
         << "CBox usable volume is "<<Volume();
   }
  // Function to calculate the volume of a CBox object
  //virtual
  double Volume() const
  { return 0.85*m_Length*m_Width*m_Height; }

  //Constructor
  CGlassBox(double lv, double wv, double hv) : CBox(lv, wv, hv){}

  // inherited from box.h as    protected     :  double m_Length;

  // inherited from box.h as    protected   :  double m_Width;

  // no access ever             private     :  double m_Height;

};
```

Since this class is ***protected*** all of the public and protected aspects of the base class now are inherited as ***protected***.

```cpp
using namespace std;
#include <iostream>

class CBox
{
 public :
 void ShowVolume() const
  { cout << endl
         << "CBox usable volume is "<<Volume();
  }
  // Function to calculate the volume of a CBox object
  //virtual
  double Volume() const
  {
    return m_Length*m_Width*m_Height; }

  //Constructor
  CBox(double lv=1.0, double wv=1.0, double hv=1.0)
                    :m_Length(lv), m_Width(wv), m_Height(hv) {}

   public      :  double m_Length;
   protected   :  double m_Width;
   private     :  double m_Height;

};
```

```cpp
#include "box.h"

class CGlassBox: private CBox
{
 public :
 void ShowVolume() const
  { cout << endl
         << "CBox usable volume is "<<Volume();
  }
  // Function to calculate the volume of a CBox object
  //virtual
  double Volume() const
  { return 0.85*m_Length*m_Width*m_Height; }

  //Constructor
  CGlassBox(double lv, double wv, double hv) : CBox(lv, wv, hv){}

  // inherited from box.h as    private      :  double m_Length;

  // inherited from box.h as    private      :  double m_Width;

  // no access ever             private      :  double m_Height;

};
```

Since this class is ***private*** all of the public and protected aspects of the base class now are inherited as ***private***.

Protected members are like private members of a class, except that they can be accessed by derived classes.

Private members can accessed on by the base class functions and not derived classes.