```
#define UNICODE
#include<windows.h>
#include<process.h>
#include"HeaderForClientOfContainmentComponentWithRegFile.h"
// global function declarations
LRESULT CALLBACK WndProc(HWND,UINT,WPARAM,LPARAM);
// global variable declarations
ISum *pISum=NULL;
ISubtract *pISubtract=NULL;
IMultiplication *pIMultiplication=NULL;
IDivision *pIDivision=NULL;
// WinMain
int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine,int nCmdShow)
{
    // variable declarations
    WNDCLASSEX wndclass;
    HWND hwnd;
    MSG msg;
    TCHAR AppName[]=TEXT("ComClient");
    HRESULT hr;
    // code
    // COM Initialization
    hr=CoInitialize(NULL);
    if(FAILED(hr))
    {
        MessageBox(NULL,TEXT("COM Library Can Not Be Initialized.\nProgram Will
          Now Exit."),TEXT("Program Error"),MB_OK);
        exit(0);
    }
    // WNDCLASSEX initialization
    wndclass.cbSize=sizeof(wndclass);
    wndclass.style=CS_HREDRAW|CS_VREDRAW;
    wndclass.cbClsExtra=0;
    wndclass.cbWndExtra=0;
    wndclass.lpfnWndProc=WndProc;
    wndclass.hIcon=LoadIcon(NULL,IDI_APPLICATION);
    wndclass.hCursor=LoadCursor(NULL,IDC_ARROW);
    wndclass.hbrBackground=(HBRUSH)GetStockObject(WHITE_BRUSH);
    wndclass.hInstance=hInstance;
    wndclass.lpszClassName=AppName;
    wndclass.lpszMenuName=NULL;
    wndclass.hIconSm=LoadIcon(NULL,IDI_APPLICATION);
    // register window class
    RegisterClassEx(&wndclass);
    // create window
    hwnd=CreateWindow(AppName,
                      TEXT("Client Of COM Dll Server"),
                      WS_OVERLAPPEDWINDOW,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
```

```
52                          CW_USEDEFAULT,
53                          NULL,
54                          NULL,
55                          hInstance,
56                          NULL);
57      ShowWindow(hwnd,nCmdShow);
58      UpdateWindow(hwnd);
59      // message loop
60      while(GetMessage(&msg,NULL,0,0))
61      {
62          TranslateMessage(&msg);
63          DispatchMessage(&msg);
64      }
65      // COM Un-initialization
66      CoUninitialize();
67      return((int)msg.wParam);
68  }
69  // Window Procedure
70  LRESULT CALLBACK WndProc(HWND hwnd,UINT iMsg,WPARAM wParam,LPARAM lParam)
71  {
72      // function declarations
73      void SafeInterfaceRelease(void);
74      // variable declarations
75      HRESULT hr;
76      int iNum1,iNum2,iSum,iSubtraction,iMultiplication,iDivision;
77      TCHAR str[255];
78      // code
79      switch(iMsg)
80      {
81      case WM_CREATE:
82          hr=CoCreateInstance(CLSID_SumSubtract,NULL,CLSCTX_INPROC_SERVER,
83                              IID_ISum,(void **)&pISum);
84          if(FAILED(hr))
85          {
86              MessageBox(hwnd,TEXT("ISum Interface Can Not Be Obtained"),TEXT
                  ("Error"),MB_OK);
87              DestroyWindow(hwnd);
88          }
89          // initialize arguments hardcoded
90          iNum1=65;
91          iNum2=45;
92          // call SumOfTwoIntegers() of ISum to get the sum
93          pISum->SumOfTwoIntegers(iNum1,iNum2,&iSum);
94          // display the result
95          wsprintf(str,TEXT("Sum Of %d And %d = %d"),iNum1,iNum2,iSum);
96          MessageBox(hwnd,str,TEXT("Result"),MB_OK);
97          // call QueryInterface() on ISum,to get ISubtract's pointer
98          hr=pISum->QueryInterface(IID_ISubtract,(void **)&pISubtract);
99          if(FAILED(hr))
100         {
101             MessageBox(hwnd,TEXT("ISubtract Interface Can Not Be Obtained"),TEXT
                  ("Error"),MB_OK);
```

```
102             DestroyWindow(hwnd);
103         }
104         // as ISum is now not needed onwords, release it
105         pISum->Release();
106         pISum=NULL;// make relesed interface NULL
107         // again initialize arguments hardcoded
108         iNum1=155;
109         iNum2=55;
110         // call SubtractionOfTwoIntegers() of ISubtract to get the subtraction
111         pISubtract->SubtractionOfTwoIntegers(iNum1,iNum2,&iSubtraction);
112         // display the result
113         wsprintf(str,TEXT("Subtraction Of %d And %d = %
              d"),iNum1,iNum2,iSubtraction);
114         MessageBox(hwnd,str,TEXT("Result"),MB_OK);
115         // call QueryInterface() on ISubtract,to get IMultiplication's pointer
116         hr=pISubtract->QueryInterface(IID_IMultiplication,(void **)
              &pIMultiplication);
117         if(FAILED(hr))
118         {
119             MessageBox(hwnd,TEXT("IMultiplication Interface Can Not Be
                  Obtained"),TEXT("Error"),MB_OK);
120             DestroyWindow(hwnd);
121         }
122         // as ISubtract is now not needed onwords, release it
123         pISubtract->Release();
124         pISubtract=NULL;// make relesed interface NULL
125         // again initialize arguments hardcoded
126         iNum1=30;
127         iNum2=25;
128         // call MultiplicationOfTwoIntegers() of IMultiplication to get the
              Multiplication
129         pIMultiplication->MultiplicationOfTwoIntegers
              (iNum1,iNum2,&iMultiplication);
130         // display the result
131         wsprintf(str,TEXT("Multiplication Of %d And %d = %
              d"),iNum1,iNum2,iMultiplication);
132         MessageBox(hwnd,str,TEXT("Result"),MB_OK);
133         // call QueryInterface() on IMultiplication's to get IDivision pointer
134         hr=pIMultiplication->QueryInterface(IID_IDivision,(void **)&pIDivision);
135         if(FAILED(hr))
136         {
137             MessageBox(hwnd,TEXT("IDivision Interface Can Not Be Obtained"),TEXT
                  ("Error"),MB_OK);
138             DestroyWindow(hwnd);
139         }
140         // as IMultiplication is now not needed onwords, release it
141         pIMultiplication->Release();
142         pIMultiplication=NULL;// make relesed interface NULL
143         // again initialize arguments hardcoded
144         iNum1=200;
145         iNum2=25;
146         // call DivisionOfTwoIntegers() of IDivision to get the Division
```

```
147         pIDivision->DivisionOfTwoIntegers(iNum1,iNum2,&iDivision);
148         // display the result
149         wsprintf(str,TEXT("Division Of %d And %d = %d"),iNum1,iNum2,iDivision);
150         MessageBox(hwnd,str,TEXT("Result"),MB_OK);
151         // finally release IDivision
152         pIDivision->Release();
153         pIDivision=NULL;// make relesed interface NULL
154         // exit the application
155         DestroyWindow(hwnd);
156         break;
157     case WM_DESTROY:
158         SafeInterfaceRelease();
159         PostQuitMessage(0);
160         break;
161     }
162     return(DefWindowProc(hwnd,iMsg,wParam,lParam));
163 }
164 void SafeInterfaceRelease(void)
165 {
166     // code
167     if(pISum)
168     {
169         pISum->Release();
170         pISum=NULL;
171     }
172     if(pISubtract)
173     {
174         pISubtract->Release();
175         pISubtract=NULL;
176     }
177     if(pIMultiplication)
178     {
179         pIMultiplication->Release();
180         pIMultiplication=NULL;
181     }
182     if(pIDivision)
183     {
184         pIDivision->Release();
185         pIDivision=NULL;
186     }
187 }
188
```

```cpp
#define UNICODE
#include<windows.h>
#include"ContainmentInnerComponentWithRegFile.h"
// class declarations
class CMultiplicationDivision:public IMultiplication,IDivision
{
private:
    long m_cRef;
public:
    // constructor method declarations
    CMultiplicationDivision(void);
    // destructor method declarations
    ~CMultiplicationDivision(void);
    // IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface(REFIID,void **);
    ULONG __stdcall AddRef(void);
    ULONG __stdcall Release(void);
    // IMultiplication specific method declarations (inherited)
    HRESULT __stdcall MultiplicationOfTwoIntegers(int,int,int *);
    // IDivision specific method declarations (inherited)
    HRESULT __stdcall DivisionOfTwoIntegers(int,int,int *);
};
class CMultiplicationDivisionClassFactory:public IClassFactory
{
private:
    long m_cRef;
public:
    // constructor method declarations
    CMultiplicationDivisionClassFactory(void);
    // destructor method declarations
    ~CMultiplicationDivisionClassFactory(void);
    // IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface(REFIID,void **);
    ULONG __stdcall AddRef(void);
    ULONG __stdcall Release(void);
    // IClassFactory specific method declarations (inherited)
    HRESULT __stdcall CreateInstance(IUnknown *,REFIID,void **);
    HRESULT __stdcall LockServer(BOOL);
};
// global variable declarations
long glNumberOfActiveComponents=0;// number of active components
long glNumberOfServerLocks=0;// number of locks on this dll
// DllMain
BOOL WINAPI DllMain(HINSTANCE hDll,DWORD dwReason,LPVOID Reserved)
{
    // code
    switch(dwReason)
    {
    case DLL_PROCESS_ATTACH:
        break;
    case DLL_PROCESS_DETACH:
        break;
```

```
53      }
54  return(TRUE);
55  }
56  // Implementation Of CMultiplicationDivision's Constructor Method
57  CMultiplicationDivision::CMultiplicationDivision(void)
58  {
59      // code
60      m_cRef=1;// hardcoded initialization to anticipate possible failure of
          QueryInterface()
61      InterlockedIncrement(&glNumberOfActiveComponents);// increment global counter
62  }
63  // Implementation Of CSumSubtract's Destructor Method
64  CMultiplicationDivision::~CMultiplicationDivision(void)
65  {
66      // code
67      InterlockedDecrement(&glNumberOfActiveComponents);// decrement global counter
68  }
69  // Implementation Of CMultiplicationDivision's IUnknown's Methods
70  HRESULT CMultiplicationDivision::QueryInterface(REFIID riid,void **ppv)
71  {
72      // code
73      if(riid==IID_IUnknown)
74          *ppv=static_cast<IMultiplication *>(this);
75      else if(riid==IID_IMultiplication)
76          *ppv=static_cast<IMultiplication *>(this);
77      else if(riid==IID_IDivision)
78          *ppv=static_cast<IDivision *>(this);
79      else
80      {
81          *ppv=NULL;
82          return(E_NOINTERFACE);
83      }
84      reinterpret_cast<IUnknown *>(*ppv)->AddRef();
85      return(S_OK);
86  }
87  ULONG CMultiplicationDivision::AddRef(void)
88  {
89      // code
90      InterlockedIncrement(&m_cRef);
91      return(m_cRef);
92  }
93  ULONG CMultiplicationDivision::Release(void)
94  {
95      // code
96      InterlockedDecrement(&m_cRef);
97      if(m_cRef==0)
98      {
99          delete(this);
100         return(0);
101     }
102     return(m_cRef);
103 }
```

```
104 // Implementation Of IMultiplication's Methods
105 HRESULT CMultiplicationDivision::MultiplicationOfTwoIntegers(int num1,int
      num2,int *pMultiplication)
106 {
107     // code
108     *pMultiplication=num1*num2;
109     return(S_OK);
110 }
111 // Implementation Of IDivision's Methods
112 HRESULT CMultiplicationDivision::DivisionOfTwoIntegers(int num1,int num2,int
      *pDivision)
113 {
114     // code
115     *pDivision=num1/num2;
116     return(S_OK);
117 }
118 // Implementation Of CMultiplicationDivisionClassFactory's Constructor Method
119 CMultiplicationDivisionClassFactory::CMultiplicationDivisionClassFactory(void)
120 {
121     // code
122     m_cRef=1;// hardcoded initialization to anticipate possible failure of
        QueryInterface()
123 }
124 // Implementation Of CMultiplicationDivisionClassFactory's Destructor Method
125 CMultiplicationDivisionClassFactory::~CMultiplicationDivisionClassFactory(void)
126 {
127     // code
128 }
129 // Implementation Of CMultiplicationDivisionClassFactory's IClassFactory's
      IUnknown's Methods
130 HRESULT CMultiplicationDivisionClassFactory::QueryInterface(REFIID riid,void
      **ppv)
131 {
132     // code
133     if(riid==IID_IUnknown)
134         *ppv=static_cast<IClassFactory *>(this);
135     else if(riid==IID_IClassFactory)
136         *ppv=static_cast<IClassFactory *>(this);
137     else
138     {
139         *ppv=NULL;
140         return(E_NOINTERFACE);
141     }
142     reinterpret_cast<IUnknown *>(*ppv)->AddRef();
143     return(S_OK);
144 }
145 ULONG CMultiplicationDivisionClassFactory::AddRef(void)
146 {
147     // code
148     InterlockedIncrement(&m_cRef);
149     return(m_cRef);
150 }
```

```
151 ULONG CMultiplicationDivisionClassFactory::Release(void)
152 {
153     // code
154     InterlockedDecrement(&m_cRef);
155     if(m_cRef==0)
156     {
157         delete(this);
158         return(0);
159     }
160     return(m_cRef);
161 }
162 // Implementation Of CMultiplicationDivisionClassFactory's IClassFactory's Methods
163 HRESULT CMultiplicationDivisionClassFactory::CreateInstance(IUnknown *pUnkOuter,REFIID riid,void **ppv)
164 {
165     // variable declarations
166     CMultiplicationDivision *pCMultiplicationDivision=NULL;
167     HRESULT hr;
168     // code
169     if(pUnkOuter!=NULL)
170         return(CLASS_E_NOAGGREGATION);
171     // create the instance of component i.e. of CMultiplicationDivision class
172     pCMultiplicationDivision=new CMultiplicationDivision;
173     if(pCMultiplicationDivision==NULL)
174         return(E_OUTOFMEMORY);
175     // get the requested interface
176     hr=pCMultiplicationDivision->QueryInterface(riid,ppv);
177     pCMultiplicationDivision->Release();// anticipate possible failure of QueryInterface()
178     return(hr);
179 }
180 HRESULT CMultiplicationDivisionClassFactory::LockServer(BOOL fLock)
181 {
182     // code
183     if(fLock)
184         InterlockedIncrement(&glNumberOfServerLocks);
185     else
186         InterlockedDecrement(&glNumberOfServerLocks);
187     return(S_OK);
188 }
189 // Implementation Of Exported Functions From This Dll
190 HRESULT __stdcall DllGetClassObject(REFCLSID rclsid,REFIID riid,void **ppv)
191 {
192     // variable declaraions
193     CMultiplicationDivisionClassFactory *pCMultiplicationDivisionClassFactory=NULL;
194     HRESULT hr;
195     // code
196     if(rclsid!=CLSID_MultiplicationDivision)
197         return(CLASS_E_CLASSNOTAVAILABLE);
198     // create class factory
```

```
199     pCMultiplicationDivisionClassFactory=new CMultiplicationDivisionClassFactory;
200     if(pCMultiplicationDivisionClassFactory==NULL)
201         return(E_OUTOFMEMORY);
202     hr=pCMultiplicationDivisionClassFactory->QueryInterface(riid,ppv);
203     pCMultiplicationDivisionClassFactory->Release();// anticipate possible
          failure of QueryInterface()
204     return(hr);
205 }
206 HRESULT __stdcall DllCanUnloadNow(void)
207 {
208     // code
209     if((glNumberOfActiveComponents==0) && (glNumberOfServerLocks==0))
210         return(S_OK);
211     else
212         return(S_FALSE);
213 }
214
```

```
1  LIBRARY ContainmentInnerComponentWithRegFile
2  EXPORTS
3        DllGetClassObject   @100 PRIVATE
4        DllCanUnloadNow     @101 PRIVATE
5
```

```
1  class IMultiplication:public IUnknown
2  {
3  public:
4      // IMultiplication specific method declarations
5      virtual HRESULT __stdcall MultiplicationOfTwoIntegers(int,int,int *)=0;// pure
          virtual
6  };
7  class IDivision:public IUnknown
8  {
9  public:
10     // IDivision specific method declarations
11     virtual HRESULT __stdcall DivisionOfTwoIntegers(int,int,int *)=0;// pure
         virtual
12 };
13 // CLSID of MultiplicationDivision Component
     {764E7C57-1737-4f19-927A-6081C39EF514}
14 const CLSID CLSID_MultiplicationDivision=
     {0x764e7c57,0x1737,0x4f19,0x92,0x7a,0x60,0x81,0xc3,0x9e,0xf5,0x14};
15 // IID of IMultiplication Interface
16 const IID IID_IMultiplication=
     {0xa39f8306,0x7a0c,0x4e47,0xb3,0x8a,0xfc,0x8d,0x68,0x5d,0xca,0x90};
17 // IID of IDivision Interface
18 const IID IID_IDivision=
     {0x9f7d9e5a,0xab6,0x4a59,0xad,0x22,0xa,0x89,0x88,0x2e,0x46,0x28};
19
```

```
REGEDIT4
[HKEY_CLASSES_ROOT\CLSID\{6D9B9F18-1DD6-4377-946A-5ED883B78031}]
@="OuterComDll_WithRegFile"
[HKEY_CLASSES_ROOT\CLSID
\{6D9B9F18-1DD6-4377-946A-5ED883B78031}\InprocServer32]
@="E:\\WINNT\\system32\\ContainmentOuterComponentWithRegFile.dll"

[HKEY_CLASSES_ROOT\CLSID\{764E7C57-1737-4f19-927A-6081C39EF514}]
@="InnerComDll_WithRegFile"
[HKEY_CLASSES_ROOT\CLSID
\{764E7C57-1737-4f19-927A-6081C39EF514}\InprocServer32]
@="E:\\WINNT\\system32\\ContainmentInnerComponentWithRegFile.dll"
```

```cpp
#define UNICODE
#include<windows.h>
#include"ContainmentInnerComponentWithRegFile.h"
#include"ContainmentOuterComponentWithRegFile.h"
// class declarations
class CSumSubtract:public ISum,ISubtract,IMultiplication,IDivision
{
private:
    long m_cRef;
    IMultiplication *m_pIMultiplication;
    IDivision *m_pIDivision;
public:
    // constructor method declarations
    CSumSubtract(void);
    // destructor method declarations
    ~CSumSubtract(void);
    // IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface(REFIID,void **);
    ULONG __stdcall AddRef(void);
    ULONG __stdcall Release(void);
    // ISum specific method declarations (inherited)
    HRESULT __stdcall SumOfTwoIntegers(int,int,int *);
    // ISubtract specific method declarations (inherited)
    HRESULT __stdcall SubtractionOfTwoIntegers(int,int,int *);
    // IMultiplication specific method declarations (inherited)
    HRESULT __stdcall MultiplicationOfTwoIntegers(int,int,int *);
    // IDivision specific method declarations (inherited)
    HRESULT __stdcall DivisionOfTwoIntegers(int,int,int *);
    // custom method for inner component creation
    HRESULT __stdcall InitializeInnerComponent(void);
};
class CSumSubtractClassFactory:public IClassFactory
{
private:
    long m_cRef;
public:
    // constructor method declarations
    CSumSubtractClassFactory(void);
    // destructor method declarations
    ~CSumSubtractClassFactory(void);
    // IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface(REFIID,void **);
    ULONG __stdcall AddRef(void);
    ULONG __stdcall Release(void);
    // IClassFactory specific method declarations (inherited)
    HRESULT __stdcall CreateInstance(IUnknown *,REFIID,void **);
    HRESULT __stdcall LockServer(BOOL);
};
// global variable declarations
long glNumberOfActiveComponents=0;// number of active components
long glNumberOfServerLocks=0;// number of locks on this dll
// DllMain
```

```
53  BOOL WINAPI DllMain(HINSTANCE hDll,DWORD dwReason,LPVOID Reserved)
54  {
55      // code
56      switch(dwReason)
57      {
58      case DLL_PROCESS_ATTACH:
59          break;
60      case DLL_PROCESS_DETACH:
61          break;
62      }
63  return(TRUE);
64  }
65  // Implementation Of CSumSubtract's Constructor Method
66  CSumSubtract::CSumSubtract(void)
67  {
68      // code
69      // initialization of private data members
70      m_pIMultiplication=NULL;
71      m_pIDivision=NULL;
72      m_cRef=1;// hardcoded initialization to anticipate possible failure of
          QueryInterface()
73      InterlockedIncrement(&glNumberOfActiveComponents);// increment global counter
74  }
75  // Implementation Of CSumSubtract's Destructor Method
76  CSumSubtract::~CSumSubtract(void)
77  {
78      // code
79      InterlockedDecrement(&glNumberOfActiveComponents);// decrement global counter
80      if(m_pIMultiplication)
81      {
82          m_pIMultiplication->Release();
83          m_pIMultiplication=NULL;
84      }
85      if(m_pIDivision)
86      {
87          m_pIDivision->Release();
88          m_pIDivision=NULL;
89      }
90  }
91  // Implementation Of CSumSubtract's IUnknown's Methods
92  HRESULT CSumSubtract::QueryInterface(REFIID riid,void **ppv)
93  {
94      // code
95      if(riid==IID_IUnknown)
96          *ppv=static_cast<ISum *>(this);
97      else if(riid==IID_ISum)
98          *ppv=static_cast<ISum *>(this);
99      else if(riid==IID_ISubtract)
100         *ppv=static_cast<ISubtract *>(this);
101     else if(riid==IID_IMultiplication)
102         *ppv=static_cast<IMultiplication *>(this);
103     else if(riid==IID_IDivision)
```

```
104         *ppv=static_cast<IDivision *>(this);
105     else
106     {
107         *ppv=NULL;
108         return(E_NOINTERFACE);
109     }
110     reinterpret_cast<IUnknown *>(*ppv)->AddRef();
111     return(S_OK);
112 }
113 ULONG CSumSubtract::AddRef(void)
114 {
115     // code
116     InterlockedIncrement(&m_cRef);
117     return(m_cRef);
118 }
119 ULONG CSumSubtract::Release(void)
120 {
121     // code
122     InterlockedDecrement(&m_cRef);
123     if(m_cRef==0)
124     {
125         delete(this);
126         return(0);
127     }
128     return(m_cRef);
129 }
130 // Implementation Of ISum's Methods
131 HRESULT CSumSubtract::SumOfTwoIntegers(int num1,int num2,int *pSum)
132 {
133     // code
134     *pSum=num1+num2;
135     return(S_OK);
136 }
137 // Implementation Of ISubtract's Methods
138 HRESULT CSumSubtract::SubtractionOfTwoIntegers(int num1,int num2,int *pSubtract)
139 {
140     // code
141     *pSubtract=num1-num2;
142     return(S_OK);
143 }
144 // Implementation Of IMultiplication's Methods
145 HRESULT CSumSubtract::MultiplicationOfTwoIntegers(int num1,int num2,int
      *pMultiplication)
146 {
147     // code
148     // delegate to inner component
149     m_pIMultiplication->MultiplicationOfTwoIntegers(num1,num2,pMultiplication);
150     return(S_OK);
151 }
152 // Implementation Of IDivision's Methods
153 HRESULT CSumSubtract::DivisionOfTwoIntegers(int num1,int num2,int *pDivision)
154 {
```

```
155     // code
156     // delegate to inner component
157     m_pIDivision->DivisionOfTwoIntegers(num1,num2,pDivision);
158     return(S_OK);
159 }
160 HRESULT CSumSubtract::InitializeInnerComponent(void)
161 {
162     // variable declarations
163     HRESULT hr;
164     // code
165     hr=CoCreateInstance(CLSID_MultiplicationDivision,NULL,CLSCTX_INPROC_SERVER,
166                         IID_IMultiplication,(void **)&m_pIMultiplication);
167     if(FAILED(hr))
168     {
169         MessageBox(NULL,TEXT("IMultiplication Interface Can Not Be Obtained From
              Inner Component."),TEXT("Error"),MB_OK);
170         return(E_FAIL);
171     }
172     hr=m_pIMultiplication->QueryInterface(IID_IDivision,(void **)&m_pIDivision);
173     if(FAILED(hr))
174     {
175         MessageBox(NULL,TEXT("IDivision Interface Can Not Be Obtained From Inner
              Component."),TEXT("Error"),MB_OK);
176         return(E_FAIL);
177     }
178     return(S_OK);
179 }
180 // Implementation Of CSumSubtractClassFactory's Constructor Method
181 CSumSubtractClassFactory::CSumSubtractClassFactory(void)
182 {
183     // code
184     m_cRef=1;// hardcoded initialization to anticipate possible failure of
          QueryInterface()
185 }
186 // Implementation Of CSumSubtractClassFactory's Destructor Method
187 CSumSubtractClassFactory::~CSumSubtractClassFactory(void)
188 {
189     // code
190 }
191 // Implementation Of CSumSubtractClassFactory's IClassFactory's IUnknown's
      Methods
192 HRESULT CSumSubtractClassFactory::QueryInterface(REFIID riid,void **ppv)
193 {
194     // code
195     if(riid==IID_IUnknown)
196         *ppv=static_cast<IClassFactory *>(this);
197     else if(riid==IID_IClassFactory)
198         *ppv=static_cast<IClassFactory *>(this);
199     else
200     {
201         *ppv=NULL;
202         return(E_NOINTERFACE);
```

```
203     }
204     reinterpret_cast<IUnknown *>(*ppv)->AddRef();
205     return(S_OK);
206 }
207 ULONG CSumSubtractClassFactory::AddRef(void)
208 {
209     // code
210     InterlockedIncrement(&m_cRef);
211     return(m_cRef);
212 }
213 ULONG CSumSubtractClassFactory::Release(void)
214 {
215     // code
216     InterlockedDecrement(&m_cRef);
217     if(m_cRef==0)
218     {
219         delete(this);
220         return(0);
221     }
222     return(m_cRef);
223 }
224 // Implementation Of CSumSubtractClassFactory's IClassFactory's Methods
225 HRESULT CSumSubtractClassFactory::CreateInstance(IUnknown *pUnkOuter,REFIID
      riid,void **ppv)
226 {
227     // variable declarations
228     CSumSubtract *pCSumSubtract=NULL;
229     HRESULT hr;
230     // code
231     if(pUnkOuter!=NULL)
232         return(CLASS_E_NOAGGREGATION);
233     // create the instance of component i.e. of CSumSubtract class
234     pCSumSubtract=new CSumSubtract;
235     if(pCSumSubtract==NULL)
236         return(E_OUTOFMEMORY);
237     // initialize the inner component
238     hr=pCSumSubtract->InitializeInnerComponent();
239     if(FAILED(hr))
240     {
241         MessageBox(NULL,TEXT("Failed To Initialize Inner Component"),TEXT
              ("Error"),MB_OK);
242         pCSumSubtract->Release();
243         return(hr);
244     }
245     // get the requested interface
246     hr=pCSumSubtract->QueryInterface(riid,ppv);
247     pCSumSubtract->Release();// anticipate possible failure of QueryInterface()
248     return(hr);
249 }
250 HRESULT CSumSubtractClassFactory::LockServer(BOOL fLock)
251 {
252     // code
```

```
253     if(fLock)
254         InterlockedIncrement(&glNumberOfServerLocks);
255     else
256         InterlockedDecrement(&glNumberOfServerLocks);
257     return(S_OK);
258 }
259 // Implementation Of Exported Functions From This Dll
260 HRESULT __stdcall DllGetClassObject(REFCLSID rclsid,REFIID riid,void **ppv)
261 {
262     // variable declaraions
263     CSumSubtractClassFactory *pCSumSubtractClassFactory=NULL;
264     HRESULT hr;
265     // code
266     if(rclsid!=CLSID_SumSubtract)
267         return(CLASS_E_CLASSNOTAVAILABLE);
268     // create class factory
269     pCSumSubtractClassFactory=new CSumSubtractClassFactory;
270     if(pCSumSubtractClassFactory==NULL)
271         return(E_OUTOFMEMORY);
272     hr=pCSumSubtractClassFactory->QueryInterface(riid,ppv);
273     pCSumSubtractClassFactory->Release();// anticipate possible failure of
        QueryInterface()
274     return(hr);
275 }
276 HRESULT __stdcall DllCanUnloadNow(void)
277 {
278     // code
279     if((glNumberOfActiveComponents==0) && (glNumberOfServerLocks==0))
280         return(S_OK);
281     else
282         return(S_FALSE);
283 }
284
```

```
1  LIBRARY ContainmentOuterComponentWithRegFile
2  EXPORTS
3        DllGetClassObject   @100 PRIVATE
4        DllCanUnloadNow     @101 PRIVATE
5
```

```
1  class ISum:public IUnknown
2  {
3  public:
4      // ISum specific method declarations
5      virtual HRESULT __stdcall SumOfTwoIntegers(int,int,int *)=0;// pure virtual
6  };
7  class ISubtract:public IUnknown
8  {
9  public:
10     // ISubtract specific method declarations
11     virtual HRESULT __stdcall SubtractionOfTwoIntegers(int,int,int *)=0;// pure
         virtual
12 };
13 // CLSID of SumSubtract Component {6D9B9F18-1DD6-4377-946A-5ED883B78031}
14 const CLSID CLSID_SumSubtract=
     {0x6d9b9f18,0x1dd6,0x4377,0x94,0x6a,0x5e,0xd8,0x83,0xb7,0x80,0x31};
15 // IID of ISum Interface
16 const IID IID_ISum=
     {0x791876b8,0x4bd,0x4202,0x91,0x8d,0xc2,0x66,0x30,0x96,0xfe,0xbf};
17 // IID of ISubtract Interface
18 const IID IID_ISubtract=
     {0x9f2a8316,0x4eda,0x4113,0xac,0x4c,0x64,0x52,0x24,0x18,0x78,0x47};
19
```

Notes For Developer .....
---

1) Inner Component Developer Will Give 2 Files, One The DLL & The Other
   Is H File.

2) Outer Component Developer Will Use Inner Component's H File In Its
   Code.
3) Outer Component Developer Will Also Have Its Own DLL & H Files.

4) During Deployment He Will Deploy Both DLLs But Not Both H Files.

5) Instead While Deploying To The Client, He Will Create Such A H File
   Which Will Encarporate Information Of Both Inner & Outer Components
   Headers Into A Seperate H File And Will Deploy This Seperate H File.
   This Seperate H File Will Contain....

   - Interface Definitions Of Both Inner & Outer Components.
   - CLSID Of Outer Component Only.
   - IID's Of All Interfaces Of Both Inner & Outer Components.

   So, If We Assume That Inner Component Is Developed By Some Other
   Vendor, The Outer Component Developer Will Behave Like That He
   Himself Developed Inner Component Too By Hiding All Information
   Of Inner Component From The Client (i.e. CLSID Of Inner Component)

   *** SPECIAL NOTE : This Seperate .H File (Which Is To Be Deployed
                      To The Client) Is Just For Deployment And Hence
                      Must Not Be Included In The Project Of Outer
                      Component Server, Though It Is Present In Its
                      Project Directory.

6) Then He Will Craete A REG File And Will Also Deploys It. This REG
   File Will Register CLSIDs & DLLs Of Both Inner & Outer Components.

7) So In All There Will Be 4 Files.....

a) Outer Component's DLL
b) Inner Component's DLL
c) Special Header File (Inner + Outer Component's H Files)
d) REG File

```
1  class ISum:public IUnknown
2  {
3  public:
4      // ISum specific method declarations
5      virtual HRESULT __stdcall SumOfTwoIntegers(int,int,int *)=0;// pure virtual
6  };
7  class ISubtract:public IUnknown
8  {
9  public:
10     // ISubtract specific method declarations
11     virtual HRESULT __stdcall SubtractionOfTwoIntegers(int,int,int *)=0;// pure
         virtual
12 };
13 class IMultiplication:public IUnknown
14 {
15 public:
16     // IMultiplication specific method declarations
17     virtual HRESULT __stdcall MultiplicationOfTwoIntegers(int,int,int *)=0;// pure
          virtual
18 };
19 class IDivision:public IUnknown
20 {
21 public:
22     // IDivision specific method declarations
23     virtual HRESULT __stdcall DivisionOfTwoIntegers(int,int,int *)=0;// pure
         virtual
24 };
25 // CLSID of SumSubtract Component {6D9B9F18-1DD6-4377-946A-5ED883B78031}
26 const CLSID CLSID_SumSubtract=
     {0x6d9b9f18,0x1dd6,0x4377,0x94,0x6a,0x5e,0xd8,0x83,0xb7,0x80,0x31};
27 // IID of ISum Interface
28 const IID IID_ISum=
     {0x791876b8,0x4bd,0x4202,0x91,0x8d,0xc2,0x66,0x30,0x96,0xfe,0xbf};
29 // IID of ISubtract Interface
30 const IID IID_ISubtract=
     {0x9f2a8316,0x4eda,0x4113,0xac,0x4c,0x64,0x52,0x24,0x18,0x78,0x47};
31 // IID of IMultiplication Interface
32 const IID IID_IMultiplication=
     {0xa39f8306,0x7a0c,0x4e47,0xb3,0x8a,0xfc,0x8d,0x68,0x5d,0xca,0x90};
33 // IID of IDivision Interface
34 const IID IID_IDivision=
     {0x9f7d9e5a,0xab6,0x4a59,0xad,0x22,0xa,0x89,0x88,0x2e,0x46,0x28};
35
```