```cpp
#define UNICODE
#include<windows.h>
#include"AggregationInnerComponentWithRegFile.h"
// interface declaration ( for internal use only. i.e. not to be included in .h file )
interface INoAggregationIUnknown // new
{
    virtual HRESULT __stdcall QueryInterface_NoAggregation(REFIID,void **)=0;
    virtual ULONG __stdcall AddRef_NoAggregation(void)=0;
    virtual ULONG __stdcall Release_NoAggregation(void)=0;
};
// class declarations
class CMultiplicationDivision:public INoAggregationIUnknown,IMultiplication,IDivision
{
private:
    long m_cRef;
    IUnknown *m_pIUnknownOuter;
public:
    // constructor method declarations
    CMultiplicationDivision(IUnknown *);// new
    // destructor method declarations
    ~CMultiplicationDivision(void);
    // Aggregation Supported IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface(REFIID,void **);
    ULONG __stdcall AddRef(void);
    ULONG __stdcall Release(void);
    // Aggregation NonSupported IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface_NoAggregation(REFIID,void **);// new
    ULONG __stdcall AddRef_NoAggregation(void);// new
    ULONG __stdcall Release_NoAggregation(void);// new
    // 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);
```

```
51  };
52  // global variable declarations
53  long glNumberOfActiveComponents=0;// number of active components
54  long glNumberOfServerLocks=0;// number of locks on this dll
55  // DllMain
56  BOOL WINAPI DllMain(HINSTANCE hDll,DWORD dwReason,LPVOID Reserved)
57  {
58      // code
59      switch(dwReason)
60      {
61      case DLL_PROCESS_ATTACH:
62          break;
63      case DLL_PROCESS_DETACH:
64          break;
65      }
66  return(TRUE);
67  }
68  // Implementation Of CMultiplicationDivision's Constructor Method
69  CMultiplicationDivision::CMultiplicationDivision(IUnknown *pIUnknownOuter)
70  {
71      // code
72      m_cRef=1;// hardcoded initialization to anticipate possible failure of
          QueryInterface()
73      InterlockedIncrement(&glNumberOfActiveComponents);// increment global counter
74      if(pIUnknownOuter!=NULL)
75          m_pIUnknownOuter=pIUnknownOuter;
76      else
77          m_pIUnknownOuter=reinterpret_cast<IUnknown *>
              (static_cast<INoAggregationIUnknown *>(this));
78  }
79  // Implementation Of CSumSubtract's Destructor Method
80  CMultiplicationDivision::~CMultiplicationDivision(void)
81  {
82      // code
83      InterlockedDecrement(&glNumberOfActiveComponents);// decrement global counter
84  }
85  // Implementation Of CMultiplicationDivision's Aggrgation Supporting IUnknown's
      Methods
86  HRESULT CMultiplicationDivision::QueryInterface(REFIID riid,void **ppv)
87  {
88      // code
89      return(m_pIUnknownOuter->QueryInterface(riid,ppv));
90  }
91  ULONG CMultiplicationDivision::AddRef(void)
92  {
93      // code
94      return(m_pIUnknownOuter->AddRef());
95  }
96  ULONG CMultiplicationDivision::Release(void)
97  {
98      // code
99      return(m_pIUnknownOuter->Release());
```

```
100 }
101 // Implementation Of CMultiplicationDivision's Aggregation NonSupporting
      IUnknown's Methods
102 HRESULT CMultiplicationDivision::QueryInterface_NoAggregation(REFIID riid,void
      **ppv)
103 {
104     // code
105     if(riid==IID_IUnknown)
106         *ppv=static_cast<INoAggregationIUnknown *>(this);
107     else if(riid==IID_IMultiplication)
108         *ppv=static_cast<IMultiplication *>(this);
109     else if(riid==IID_IDivision)
110         *ppv=static_cast<IDivision *>(this);
111     else
112     {
113         *ppv=NULL;
114         return(E_NOINTERFACE);
115     }
116     reinterpret_cast<IUnknown *>(*ppv)->AddRef();
117     return(S_OK);
118 }
119 ULONG CMultiplicationDivision::AddRef_NoAggregation(void)
120 {
121     // code
122     InterlockedIncrement(&m_cRef);
123     return(m_cRef);
124 }
125 ULONG CMultiplicationDivision::Release_NoAggregation(void)
126 {
127     // code
128     InterlockedDecrement(&m_cRef);
129     if(m_cRef==0)
130     {
131         delete(this);
132         return(0);
133     }
134     return(m_cRef);
135 }
136 // Implementation Of IMultiplication's Methods
137 HRESULT CMultiplicationDivision::MultiplicationOfTwoIntegers(int num1,int
      num2,int *pMultiplication)
138 {
139     // code
140     *pMultiplication=num1*num2;
141     return(S_OK);
142 }
143 // Implementation Of IDivision's Methods
144 HRESULT CMultiplicationDivision::DivisionOfTwoIntegers(int num1,int num2,int
      *pDivision)
145 {
146     // code
147     *pDivision=num1/num2;
```

```
148     return(S_OK);
149 }
150 // Implementation Of CMultiplicationDivisionClassFactory's Constructor Method
151 CMultiplicationDivisionClassFactory::CMultiplicationDivisionClassFactory(void)
152 {
153     // code
154     m_cRef=1;// hardcoded initialization to anticipate possible failure of
          QueryInterface()
155 }
156 // Implementation Of CMultiplicationDivisionClassFactory's Destructor Method
157 CMultiplicationDivisionClassFactory::~CMultiplicationDivisionClassFactory(void)
158 {
159     // code
160 }
161 // Implementation Of CMultiplicationDivisionClassFactory's IClassFactory's
      IUnknown's Methods
162 HRESULT CMultiplicationDivisionClassFactory::QueryInterface(REFIID riid,void
      **ppv)
163 {
164     // code
165     if(riid==IID_IUnknown)
166         *ppv=static_cast<IClassFactory *>(this);
167     else if(riid==IID_IClassFactory)
168         *ppv=static_cast<IClassFactory *>(this);
169     else
170     {
171         *ppv=NULL;
172         return(E_NOINTERFACE);
173     }
174     reinterpret_cast<IUnknown *>(*ppv)->AddRef();
175     return(S_OK);
176 }
177 ULONG CMultiplicationDivisionClassFactory::AddRef(void)
178 {
179     // code
180     InterlockedIncrement(&m_cRef);
181     return(m_cRef);
182 }
183 ULONG CMultiplicationDivisionClassFactory::Release(void)
184 {
185     // code
186     InterlockedDecrement(&m_cRef);
187     if(m_cRef==0)
188     {
189         delete(this);
190         return(0);
191     }
192     return(m_cRef);
193 }
194 // Implementation Of CMultiplicationDivisionClassFactory's IClassFactory's
      Methods
195 HRESULT CMultiplicationDivisionClassFactory::CreateInstance(IUnknown
```

```
    *pUnkOuter,REFIID riid,void **ppv)
{
    // variable declarations
    CMultiplicationDivision *pCMultiplicationDivision=NULL;
    HRESULT hr;
    // code
    if((pUnkOuter!=NULL) && (riid!=IID_IUnknown))
        return(CLASS_E_NOAGGREGATION);
    // create the instance of component i.e. of CMultiplicationDivision class
    pCMultiplicationDivision=new CMultiplicationDivision(pUnkOuter);
    if(pCMultiplicationDivision==NULL)
        return(E_OUTOFMEMORY);
    // get the requested interface
    hr=pCMultiplicationDivision->QueryInterface_NoAggregation(riid,ppv);
    pCMultiplicationDivision->Release_NoAggregation();// anticipate possible
      failure of QueryInterface()
    return(hr);
}
HRESULT CMultiplicationDivisionClassFactory::LockServer(BOOL fLock)
{
    // code
    if(fLock)
        InterlockedIncrement(&glNumberOfServerLocks);
    else
        InterlockedDecrement(&glNumberOfServerLocks);
    return(S_OK);
}
// Implementation Of Exported Functions From This Dll
HRESULT __stdcall DllGetClassObject(REFCLSID rclsid,REFIID riid,void **ppv)
{
    // variable declaraions
    CMultiplicationDivisionClassFactory
      *pCMultiplicationDivisionClassFactory=NULL;
    HRESULT hr;
    // code
    if(rclsid!=CLSID_MultiplicationDivision)
        return(CLASS_E_CLASSNOTAVAILABLE);
    // create class factory
    pCMultiplicationDivisionClassFactory=new CMultiplicationDivisionClassFactory;
    if(pCMultiplicationDivisionClassFactory==NULL)
        return(E_OUTOFMEMORY);
    hr=pCMultiplicationDivisionClassFactory->QueryInterface(riid,ppv);
    pCMultiplicationDivisionClassFactory->Release();// anticipate possible
      failure of QueryInterface()
    return(hr);
}
HRESULT __stdcall DllCanUnloadNow(void)
{
    // code
    if((glNumberOfActiveComponents==0) && (glNumberOfServerLocks==0))
        return(S_OK);
    else
```

```
244             return(S_FALSE);
245     }
246
```

```
1  LIBRARY AggregationInnerComponentWithRegFile
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 {4AC37EEC-DAFD-435d-
     A1BC-261EFA28EF46}
14 const CLSID CLSID_MultiplicationDivision=
     {0x4ac37eec,0xdafd,0x435d,0xa1,0xbc,0x26,0x1e,0xfa,0x28,0xef,0x46};
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\{FA5EC586-38C0-4d05-9744-A52D4B13292A}]
@="OuterComDll_WithRegFile"
[HKEY_CLASSES_ROOT\CLSID\{FA5EC586-38C0-4d05-9744-
A52D4B13292A}\InprocServer32]
@="E:\\WINNT\\system32\\AggregationOuterComponentWithRegFile.dll"

[HKEY_CLASSES_ROOT\CLSID\{4AC37EEC-DAFD-435d-A1BC-261EFA28EF46}]
@="InnerComDll_WithRegFile"
[HKEY_CLASSES_ROOT\CLSID\{4AC37EEC-DAFD-435d-
A1BC-261EFA28EF46}\InprocServer32]
@="E:\\WINNT\\system32\\AggregationInnerComponentWithRegFile.dll"
```

```cpp
#define UNICODE
#include<windows.h>
#include"AggregationInnerComponentWithRegFile.h"
#include"AggregationOuterComponentWithRegFile.h"
// class declarations
class CSumSubtract:public ISum,ISubtract
{
private:
    long m_cRef;
    IUnknown *m_pIUnknownInner;
    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 *);
    // 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
BOOL WINAPI DllMain(HINSTANCE hDll,DWORD dwReason,LPVOID Reserved)
{
    // code
```

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

```
104     else if(riid==IID_IMultiplication)
105         return(m_pIUnknownInner->QueryInterface(riid,ppv));
106     else if(riid==IID_IDivision)
107         return(m_pIUnknownInner->QueryInterface(riid,ppv));
108     else
109     {
110         *ppv=NULL;
111         return(E_NOINTERFACE);
112     }
113     reinterpret_cast<IUnknown *>(*ppv)->AddRef();
114     return(S_OK);
115 }
116 ULONG CSumSubtract::AddRef(void)
117 {
118     // code
119     InterlockedIncrement(&m_cRef);
120     return(m_cRef);
121 }
122 ULONG CSumSubtract::Release(void)
123 {
124     // code
125     InterlockedDecrement(&m_cRef);
126     if(m_cRef==0)
127     {
128         delete(this);
129         return(0);
130     }
131     return(m_cRef);
132 }
133 // Implementation Of ISum's Methods
134 HRESULT CSumSubtract::SumOfTwoIntegers(int num1,int num2,int *pSum)
135 {
136     // code
137     *pSum=num1+num2;
138     return(S_OK);
139 }
140 // Implementation Of ISubtract's Methods
141 HRESULT CSumSubtract::SubtractionOfTwoIntegers(int num1,int num2,int *pSubtract)
142 {
143     // code
144     *pSubtract=num1-num2;
145     return(S_OK);
146 }
147 HRESULT CSumSubtract::InitializeInnerComponent(void)
148 {
149     // variable declarations
150     HRESULT hr;
151     // code
152     hr=CoCreateInstance(CLSID_MultiplicationDivision,
153                         reinterpret_cast<IUnknown *>(this),
154                         CLSCTX_INPROC_SERVER,
155                         IID_IUnknown,
```

```
156                             (void **)&m_pIUnknownInner);
157     if(FAILED(hr))
158     {
159         MessageBox(NULL,TEXT("IUnknown Interface Can Not Be Obtained From Inner
              Component."),TEXT("Error"),MB_OK);
160         return(E_FAIL);
161     }
162     hr=m_pIUnknownInner->QueryInterface(IID_IMultiplication,(void **)
          &m_pIMultiplication);
163     if(FAILED(hr))
164     {
165         MessageBox(NULL,TEXT("IMultiplication Interface Can Not Be Obtained From
              Inner Component."),TEXT("Error"),MB_OK);
166         m_pIUnknownInner->Release();
167         m_pIUnknownInner=NULL;
168         return(E_FAIL);
169     }
170     hr=m_pIUnknownInner->QueryInterface(IID_IDivision,(void **)&m_pIDivision);
171     if(FAILED(hr))
172     {
173         MessageBox(NULL,TEXT("IDivision Interface Can Not Be Obtained From Inner
              Component."),TEXT("Error"),MB_OK);
174         m_pIUnknownInner->Release();
175         m_pIUnknownInner=NULL;
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 AggregationOuterComponentWithRegFile
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 {FA5EC586-38C0-4d05-9744-A52D4B13292A}
14 const CLSID CLSID_SumSubtract={0xfa5ec586, 0x38c0, 0x4d05, 0x97, 0x44, 0xa5, 0x2d,
     0x4b, 0x13, 0x29, 0x2a};
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
```

```cpp
#define UNICODE
#include<windows.h>
#include"HeaderForClientOfAggregationComponentWithRegFile.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,
                      CW_USEDEFAULT,
```

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

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

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

```
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 {FA5EC586-38C0-4d05-9744-A52D4B13292A}
26 const CLSID CLSID_SumSubtract={0xfa5ec586, 0x38c0, 0x4d05, 0x97, 0x44, 0xa5, 0x2d,
     0x4b, 0x13, 0x29, 0x2a};
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 // CLSID of MultiplicationDivision Component {4AC37EEC-DAFD-435d-
     A1BC-261EFA28EF46}
32 const CLSID CLSID_MultiplicationDivision=
     {0x4ac37eec,0xdafd,0x435d,0xa1,0xbc,0x26,0x1e,0xfa,0x28,0xef,0x46};
33 // IID of IMultiplication Interface
34 const IID IID_IMultiplication=
     {0xa39f8306,0x7a0c,0x4e47,0xb3,0x8a,0xfc,0x8d,0x68,0x5d,0xca,0x90};
35 // IID of IDivision Interface
36 const IID IID_IDivision=
     {0x9f7d9e5a,0xab6,0x4a59,0xad,0x22,0xa,0x89,0x88,0x2e,0x46,0x28};
37
```