

```

1  #define UNICODE
2  #include<windows.h>
3  #include"AggregationInnerComponentWithRegFile.h"
4  // interface declaration ( for internal use only. i.e. not to be included in .h file )
5  interface INoAggregationIUnknown // new
6  {
7      virtual HRESULT __stdcall QueryInterface_NoAggregation(REFIID,void **)=0;
8      virtual ULONG __stdcall AddRef_NoAggregation(void)=0;
9      virtual ULONG __stdcall Release_NoAggregation(void)=0;
10 };
11 // class declarations
12 class CMultiplicationDivision:public INoAggregationIUnknown,IMultiplication,IDivision
13 {
14 private:
15     long m_cRef;
16     IUnknown *m_pIUnknownOuter;
17 public:
18     // constructor method declarations
19     CMultiplicationDivision(IUnknown *);// new
20     // destructor method declarations
21     ~CMultiplicationDivision(void);
22     // Aggregation Supported IUnknown specific method declarations (inherited)
23     HRESULT __stdcall QueryInterface(REFIID,void **);
24     ULONG __stdcall AddRef(void);
25     ULONG __stdcall Release(void);
26     // Aggregation NonSupported IUnknown specific method declarations (inherited)
27     HRESULT __stdcall QueryInterface_NoAggregation(REFIID,void **);// new
28     ULONG __stdcall AddRef_NoAggregation(void);// new
29     ULONG __stdcall Release_NoAggregation(void);// new
30     // IMultiplication specific method declarations (inherited)
31     HRESULT __stdcall MultiplicationOfTwoIntegers(int,int,int *);
32     // IDivision specific method declarations (inherited)
33     HRESULT __stdcall DivisionOfTwoIntegers(int,int,int *);
34 };
35 class CMultiplicationDivisionClassFactory:public IClassFactory
36 {
37 private:
38     long m_cRef;
39 public:
40     // constructor method declarations
41     CMultiplicationDivisionClassFactory(void);
42     // destructor method declarations
43     ~CMultiplicationDivisionClassFactory(void);
44     // IUnknown specific method declarations (inherited)
45     HRESULT __stdcall QueryInterface(REFIID,void **);
46     ULONG __stdcall AddRef(void);
47     ULONG __stdcall Release(void);
48     // IClassFactory specific method declarations (inherited)
49     HRESULT __stdcall CreateInstance(IUnknown *,REFIID,void **);
50     HRESULT __stdcall LockServer(BOOL);

```

```

51 };
52 // global variable declarations
53 long gINumberOfActiveComponents=0;// number of active components
54 long gINumberOfServerLocks=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
73     QueryInterface()
74     InterlockedIncrement(&gINumberOfActiveComponents);// increment global counter
75     if(pIUnknownOuter!=NULL)
76         m_pIUnknownOuter=pIUnknownOuter;
77     else
78         m_pIUnknownOuter=reinterpret_cast<IUnknown *>
79         (static_cast<INoAggregationIUnknown *>(this));
80 }
81 // Implementation Of CSumSubtract's Destructor Method
82 CMultiplicationDivision::~CMultiplicationDivision(void)
83 {
84     // code
85     InterlockedDecrement(&gINumberOfActiveComponents);// decrement global counter
86 }
87 // Implementation Of CMultiplicationDivision's Aggragation Supporting IUnknown's
88 // Methods
89 HRESULT CMultiplicationDivision::QueryInterface(REFIID riid,void **ppv)
90 {
91     // code
92     return(m_pIUnknownOuter->QueryInterface(riid,ppv));
93 }
94 ULONG CMultiplicationDivision::AddRef(void)
95 {
96     // code
97     return(m_pIUnknownOuter->AddRef());
98 }
99 ULONG CMultiplicationDivision::Release(void)
100 {
101     // code
102     return(m_pIUnknownOuter->Release());
103 }

```

```
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)
196 {
197     // variable declarations
198     CMultiplicationDivision *pCMultiplicationDivision=NULL;
199     HRESULT hr;
200     // code
201     if((pUnkOuter!=NULL) && (riid!=IID_IUnknown))
202         return(CLASS_E_NOAGGREGATION);
203     // create the instance of component i.e. of CMultiplicationDivision class
204     pCMultiplicationDivision=new CMultiplicationDivision(pUnkOuter);
205     if(pCMultiplicationDivision==NULL)
206         return(E_OUTOFMEMORY);
207     // get the requested interface
208     hr=pCMultiplicationDivision->QueryInterface_NoAggregation(riid,ppv);
209     pCMultiplicationDivision->Release_NoAggregation();// anticipate possible failure of QueryInterface()
210     return(hr);
211 }
212 HRESULT CMultiplicationDivisionClassFactory::LockServer(BOOL fLock)
213 {
214     // code
215     if(fLock)
216         InterlockedIncrement(&glNumberOfServerLocks);
217     else
218         InterlockedDecrement(&glNumberOfServerLocks);
219     return(S_OK);
220 }
221 // Implementation Of Exported Functions From This Dll
222 HRESULT __stdcall DllGetClassObject(REFCLSID rclsid, REFIID riid, void **ppv)
223 {
224     // variable declarations
225     CMultiplicationDivisionClassFactory *pCMultiplicationDivisionClassFactory=NULL;
226     HRESULT hr;
227     // code
228     if(rclsid!=CLSID_MultiplicationDivision)
229         return(CLASS_E_CLASSNOTAVAILABLE);
230     // create class factory
231     pCMultiplicationDivisionClassFactory=new CMultiplicationDivisionClassFactory;
232     if(pCMultiplicationDivisionClassFactory==NULL)
233         return(E_OUTOFMEMORY);
234     hr=pCMultiplicationDivisionClassFactory->QueryInterface(riid,ppv);
235     pCMultiplicationDivisionClassFactory->Release();// anticipate possible failure of QueryInterface()
236     return(hr);
237 }
238 HRESULT __stdcall DllCanUnloadNow(void)
239 {
240     // code
241     if((glNumberOfActiveComponents==0) && (glNumberOfServerLocks==0))
242         return(S_OK);
243     else

```

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