

```
1 #include<windows.h>
2 #include"AggregationInnerComponentWithRegFile.h"
3 #include"AggregationOuterComponentWithRegFile.h"
4 // class declarations
5 class CSumSubtract:public ISum,ISubtract
6 {
7     private:
8         long m_cRef;
9         IUnknown *m_pIUnknownInner;
10        IMultiplication *m_pIMultiplication;
11        IDivision *m_pIDivision;
12    public:
13        // constructor method declarations
14        CSumSubtract(void);
15        // destructor method declarations
16        ~CSumSubtract(void);
17        // IUnknown specific method declarations (inherited)
18        HRESULT __stdcall QueryInterface(REFIID,void **);
19        ULONG __stdcall AddRef(void);
20        ULONG __stdcall Release(void);
21        // ISum specific method declarations (inherited)
22        HRESULT __stdcall SumOfTwoIntegers(int,int,int *);
23        // ISubtract specific method declarations (inherited)
24        HRESULT __stdcall SubtractionOfTwoIntegers(int,int,int *);
25        // custom method for inner component creation
26        HRESULT __stdcall InitializeInnerComponent(void);
27    };
28    class CSumSubtractClassFactory:public IClassFactory
29    {
30        private:
31            long m_cRef;
32        public:
33            // constructor method declarations
34            CSumSubtractClassFactory(void);
35            // destructor method declarations
36            ~CSumSubtractClassFactory(void);
37            // IUnknown specific method declarations (inherited)
38            HRESULT __stdcall QueryInterface(REFIID,void **);
39            ULONG __stdcall AddRef(void);
40            ULONG __stdcall Release(void);
41            // IClassFactory specific method declarations (inherited)
42            HRESULT __stdcall CreateInstance(IUnknown *,REFIID,void **);
43            HRESULT __stdcall LockServer(BOOL);
44    };
45    // global variable declarations
46    long glNumberOfActiveComponents=0;// number of active components
47    long glNumberOfServerLocks=0;// number of locks on this dll
48    // DllMain
49    BOOL WINAPI DllMain(HINSTANCE hDll,WORD dwReason,LPVOID Reserved)
50    {
51        // 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 ↵
71     QueryInterface()
72     InterlockedIncrement(&g_lNumberOfActiveComponents);// increment global counter
73 }
74 // Implementation Of CSumSubtract's Destructor Method
75 CSumSubtract::~CSumSubtract(void)
76 {
77     // code
78     InterlockedDecrement(&g_lNumberOfActiveComponents);// decrement global counter
79     if(m_pIMultiplication)
80     {
81         m_pIMultiplication->Release();
82         m_pIMultiplication=NULL;
83     }
84     if(m_pIDivision)
85     {
86         m_pIDivision->Release();
87         m_pIDivision=NULL;
88     }
89     if(m_pIUnknownInner)
90     {
91         m_pIUnknownInner->Release();
92         m_pIUnknownInner=NULL;
93     }
94 }
95 // Implementation Of CSumSubtract's IUnknown's Methods
96 HRESULT CSumSubtract::QueryInterface(REFIID riid,void **ppv)
97 {
98     // code
99     if(riid==IID_IUnknown)
100         *ppv=static_cast<ISum *>(this);
101     else if(riid==IID_ISum)
102         *ppv=static_cast<ISum *>(this);
103     else if(riid==IID_ISubtract)
104         *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 }
// Implementation Of CSumSubtractClassFactory's Constructor Method
180 CSumSubtractClassFactory::CSumSubtractClassFactory(void)
181 {
182     // code
183     m_cRef=1;// hardcoded initialization to anticipate possible failure of QueryInterface()
184 }
// Implementation Of CSumSubtractClassFactory's Destructor Method
185 CSumSubtractClassFactory::~CSumSubtractClassFactory(void)
186 {
187     // code
188 }
// Implementation Of CSumSubtractClassFactory's IClassFactory's IUnknown's Methods
189 HRESULT CSumSubtractClassFactory::QueryInterface(REFIID riid,void **ppv)
190 {
191     // code
192     if(riid==IID_IUnknown)
193         *ppv=static_cast(this);
194     else if(riid==IID_IClassFactory)
195         *ppv=static_cast(this);
196     else
197     {
198         *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
242             ("Error"),MB_OK);
243         pCSumSubtract->Release();
244         return(hr);
245     }
246     // get the requested interface
247     hr=pCSumSubtract->QueryInterface(rapidjson::riid,ppv);
248     pCSumSubtract->Release(); // anticipate possible failure of QueryInterface()
249     return(hr);
250 }
251 HRESULT CSumSubtractClassFactory::LockServer(BOOL fLock)
```

```
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 declarations
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
274     QueryInterface()
275     return(hr);
276 }
277 HRESULT __stdcall DllCanUnloadNow(void)
278 {
279     // code
280     if((glNumberOfActiveComponents==0) && (glNumberOfServerLocks==0))
281         return(S_OK);
282     else
283         return(S_FALSE);
284 }
```