```cpp

#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
```