```cpp
// Header Files
#include <windows.h>
#include <stdio.h> // for swprintf_s()
#include "AutomationServerWithRegFile.h"

// class declarations
// coclass
class CMyMath :public IMyMath
{
private:
    long m_cRef;
    ITypeInfo *m_pITypeInfo;
public:
    // constructor method declarations
    CMyMath(void);
    // destructor method declarations
    ~CMyMath(void);
    // IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface(REFIID, void **);
    ULONG __stdcall AddRef(void);
    ULONG __stdcall Release(void);
    // IDispatch specific method declarations (inherited)
    HRESULT __stdcall GetTypeInfoCount(UINT*);
    HRESULT __stdcall GetTypeInfo(UINT, LCID, ITypeInfo**);
    HRESULT __stdcall GetIDsOfNames(REFIID, LPOLESTR*, UINT, LCID, DISPID*);
    HRESULT __stdcall Invoke(DISPID, REFIID, LCID, WORD, DISPPARAMS*,
      VARIANT*, EXCEPINFO*, UINT*);
    // ISum specific method declarations (inherited)
    HRESULT __stdcall SumOfTwoIntegers(int, int, int *);
    // ISubtract specific method declarations (inherited)
    HRESULT __stdcall SubtractionOfTwoIntegers(int, int, int *);
    // custom methods
    HRESULT InitInstance(void);
};

// class factory
class CMyMathClassFactory :public IClassFactory
{
private:
    long m_cRef;
public:
    // constructor method declarations
    CMyMathClassFactory(void);
    // destructor method declarations
    ~CMyMathClassFactory(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);
};
```

```
53
54  // global variable declarations
55  long glNumberOfActiveComponents = 0;// number of active components
56  long glNumberOfServerLocks = 0;// number of locks on this dll
57  // {1F879C17-26BE-420C-A3F6-2995E0970AF3}
58  const GUID LIBID_AutomationServer = { 0x1f879c17, 0x26be, 0x420c, 0xa3,
      0xf6, 0x29, 0x95, 0xe0, 0x97, 0xa, 0xf3 };
59
60  // DllMain
61  BOOL WINAPI DllMain(HINSTANCE hDll, DWORD dwReason, LPVOID Reserved)
62  {
63      // code
64      switch (dwReason)
65      {
66      case DLL_PROCESS_ATTACH:
67          break;
68      case DLL_PROCESS_DETACH:
69          break;
70      }
71      return(TRUE);
72  }
73
74  // Implementation Of CMyMath's Constructor Method
75  CMyMath::CMyMath(void)
76  {
77      // code
78      m_cRef = 1;// hardcoded initialization to anticipate possible failure of
          QueryInterface()
79      InterlockedIncrement(&glNumberOfActiveComponents);// increment global
        counter
80  }
81
82  // Implementation Of CMyMath's Destructor Method
83  CMyMath::~CMyMath(void)
84  {
85      // code
86      InterlockedDecrement(&glNumberOfActiveComponents);// decrement global
        counter
87  }
88
89  // Implementation Of CMyMath's IUnknown's Methods
90  HRESULT CMyMath::QueryInterface(REFIID riid, void **ppv)
91  {
92      // code
93      if (riid == IID_IUnknown)
94          *ppv = static_cast<IMyMath *>(this);
95      else if (riid == IID_IDispatch)
96          *ppv = static_cast<IMyMath *>(this);
97      else if (riid == IID_IMyMath)
98          *ppv = static_cast<IMyMath *>(this);
99      else
100     {
101         *ppv = NULL;
```

```
102         return(E_NOINTERFACE);
103     }
104     reinterpret_cast<IUnknown *>(*ppv)->AddRef();
105     return(S_OK);
106 }
107
108 ULONG CMyMath::AddRef(void)
109 {
110     // code
111     InterlockedIncrement(&m_cRef);
112     return(m_cRef);
113 }
114
115 ULONG CMyMath::Release(void)
116 {
117     // code
118     InterlockedDecrement(&m_cRef);
119     if (m_cRef == 0)
120     {
121         m_pITypeInfo->Release();
122         m_pITypeInfo = NULL;
123         delete(this);
124         return(0);
125     }
126     return(m_cRef);
127 }
128
129 // Implementation Of IMyMath's Methods
130 HRESULT CMyMath::SumOfTwoIntegers(int num1, int num2, int *pSum)
131 {
132     // code
133     *pSum = num1 + num2;
134     return(S_OK);
135 }
136
137 HRESULT CMyMath::SubtractionOfTwoIntegers(int num1, int num2, int
      *pSubtract)
138 {
139     // code
140     *pSubtract = num1 - num2;
141     return(S_OK);
142 }
143
144 HRESULT CMyMath::InitInstance(void)
145 {
146     // function declarations
147     void ComErrorDescriptionString(HWND, HRESULT);
148     // variable declarations
149     HRESULT hr;
150     ITypeLib *pITypeLib = NULL;
151     // code
152     if (m_pITypeInfo == NULL)
153     {
```

```
154         hr = LoadRegTypeLib(LIBID_AutomationServer,
155              1, 0,// major/minor version numbers
156              0x00,// LANG_NEUTRAL
157              &pITypeLib);
158         if (FAILED(hr))
159         {
160             ComErrorDescriptionString(NULL, hr);
161             return(hr);
162         }
163         hr = pITypeLib->GetTypeInfoOfGuid(IID_IMyMath, &m_pITypeInfo);
164         if (FAILED(hr))
165         {
166             ComErrorDescriptionString(NULL, hr);
167             pITypeLib->Release();
168             return(hr);
169         }
170         pITypeLib->Release();
171     }
172     return(S_OK);
173 }
174
175 // Implementation Of CMyMathClassFactory's Constructor Method
176 CMyMathClassFactory::CMyMathClassFactory(void)
177 {
178     // code
179     m_cRef = 1;// hardcoded initialization to anticipate possible failure of ↵
        QueryInterface()
180 }
181
182 // Implementation Of CMyMathClassFactory's Destructor Method
183 CMyMathClassFactory::~CMyMathClassFactory(void)
184 {
185     // code
186 }
187
188 // Implementation Of CMyMathClassFactory's IClassFactory's IUnknown's        ↵
      Methods
189 HRESULT CMyMathClassFactory::QueryInterface(REFIID riid, void **ppv)
190 {
191     // code
192     if (riid == IID_IUnknown)
193         *ppv = static_cast<IClassFactory *>(this);
194     else if (riid == IID_IClassFactory)
195         *ppv = static_cast<IClassFactory *>(this);
196     else
197     {
198         *ppv = NULL;
199         return(E_NOINTERFACE);
200     }
201     reinterpret_cast<IUnknown *>(*ppv)->AddRef();
202     return(S_OK);
203 }
204
```

```
205 ULONG CMyMathClassFactory::AddRef(void)
206 {
207     // code
208     InterlockedIncrement(&m_cRef);
209     return(m_cRef);
210 }
211 ULONG CMyMathClassFactory::Release(void)
212 {
213     // code
214     InterlockedDecrement(&m_cRef);
215     if (m_cRef == 0)
216     {
217         delete(this);
218         return(0);
219     }
220     return(m_cRef);
221 }
222
223 // Implementation Of CMyMathClassFactory's IClassFactory's Methods
224 HRESULT CMyMathClassFactory::CreateInstance(IUnknown *pUnkOuter, REFIID
      riid, void **ppv)
225 {
226     // variable declarations
227     CMyMath *pCMyMath = NULL;
228     HRESULT hr;
229     // code
230     if (pUnkOuter != NULL)
231         return(CLASS_E_NOAGGREGATION);
232     // create the instance of component i.e. of CMyMath class
233     pCMyMath = new CMyMath;
234     if (pCMyMath == NULL)
235         return(E_OUTOFMEMORY);
236
237     // call automation related init method
238     pCMyMath->InitInstance();
239
240     // get the requested interface
241     hr = pCMyMath->QueryInterface(riid, ppv);
242     pCMyMath->Release();// anticipate possible failure of QueryInterface()
243     return(hr);
244 }
245
246 HRESULT CMyMathClassFactory::LockServer(BOOL fLock)
247 {
248     // code
249     if (fLock)
250         InterlockedIncrement(&glNumberOfServerLocks);
251     else
252         InterlockedDecrement(&glNumberOfServerLocks);
253     return(S_OK);
254 }
255
256 // Implementation of IDispatch's methods
```

```
257 HRESULT CMyMath::GetTypeInfoCount(UINT *pCountTypeInfo)
258
259 {
260     // code
261     // as we have type library it is 1, else 0
262     *pCountTypeInfo = 1;
263     return(S_OK);
264 }
265
266 HRESULT CMyMath::GetTypeInfo(UINT iTypeInfo, LCID lcid, ITypeInfo
      **ppITypeInfo)
267 {
268     // code
269     *ppITypeInfo = NULL;
270     if (iTypeInfo != 0)
271         return(DISP_E_BADINDEX);
272     m_pITypeInfo->AddRef();
273     *ppITypeInfo = m_pITypeInfo;
274     return(S_OK);
275 }
276
277 HRESULT CMyMath::GetIDsOfNames(REFIID riid, LPOLESTR *rgszNames, UINT
      cNames, LCID lcid, DISPID *rgDispId)
278 {
279     // code
280     return(DispGetIDsOfNames(m_pITypeInfo, rgszNames, cNames, rgDispId));
281 }
282
283 HRESULT CMyMath::Invoke(DISPID dispIdMember, REFIID riid, LCID lcid, WORD
      wFlags, DISPPARAMS *pDispParams, VARIANT *pVarResult, EXCEPINFO
      *pExcepInfo, UINT *puArgErr)
284 {
285     // variable declarations
286     HRESULT hr;
287     // code
288     hr = DispInvoke(this,
289         m_pITypeInfo,
290         dispIdMember,
291         wFlags,
292         pDispParams,
293         pVarResult,
294         pExcepInfo,
295         puArgErr);
296     return(hr);
297 }
298
299 // Implementation Of Exported Functions From This Dll
300 extern "C" HRESULT __stdcall DllGetClassObject(REFCLSID rclsid, REFIID riid,
      void **ppv)
301 {
302     // variable declaraions
303     CMyMathClassFactory *pCMyMathClassFactory = NULL;
304     HRESULT hr;
```

```
305     // code
306     if (rclsid != CLSID_MyMath)
307         return(CLASS_E_CLASSNOTAVAILABLE);
308     // create class factory
309     pCMyMathClassFactory = new CMyMathClassFactory;
310     if (pCMyMathClassFactory == NULL)
311         return(E_OUTOFMEMORY);
312     hr = pCMyMathClassFactory->QueryInterface(riid, ppv);
313     pCMyMathClassFactory->Release();// anticipate possible failure of
          QueryInterface()
314     return(hr);
315 }
316
317 extern "C" HRESULT __stdcall DllCanUnloadNow(void)
318 {
319     // code
320     if ((glNumberOfActiveComponents == 0) && (glNumberOfServerLocks == 0))
321         return(S_OK);
322     else
323         return(S_FALSE);
324 }
325
326 void ComErrorDescriptionString(HWND hwnd, HRESULT hr)
327 {
328     // variable declarations
329     TCHAR* szErrorMessage = NULL;
330     TCHAR str[255];
331     // code
332     if (FACILITY_WINDOWS == HRESULT_FACILITY(hr))
333         hr = HRESULT_CODE(hr);
334
335     if (FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |
          FORMAT_MESSAGE_FROM_SYSTEM, NULL, hr, MAKELANGID(LANG_NEUTRAL,
          SUBLANG_DEFAULT), (LPTSTR)&szErrorMessage, 0, NULL) != 0)
336     {
337         swprintf_s(str, TEXT("%#x : %s"), hr,szErrorMessage);
338         LocalFree(szErrorMessage);
339     }
340     else
341         swprintf_s(str, TEXT("[Could not find a description for error # %
              #x.]\n"), hr);
342
343     MessageBox(hwnd, str, TEXT("COM Error"), MB_OK);
344 }
345
```