```
#include<windows.h>
#include <stdio.h> // for swprintf_s()
#include"AutomationServerWithRegFile.h"
// global function declarations
LRESULT CALLBACK WndProc(HWND,UINT,WPARAM,LPARAM);
// WinMain
int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine,int nCmdShow)
{
    // variable declarations
    WNDCLASSEX wndclass;
    HWND hwnd;
    MSG msg;
    TCHAR AppName[]=TEXT("Client");
    // code
    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 Exe Server"),
                      WS_OVERLAPPEDWINDOW,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      NULL,
                      NULL,
                      hInstance,
                      NULL);
    ShowWindow(hwnd,nCmdShow);
    UpdateWindow(hwnd);
    // message loop
    while(GetMessage(&msg,NULL,0,0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return((int)msg.wParam);
}
// Window Procedure
LRESULT CALLBACK WndProc(HWND hwnd,UINT iMsg,WPARAM wParam,LPARAM lParam)
```

```
54  {
55      // function declarations
56      void ComErrorDescriptionString(HWND, HRESULT);
57      // variable declarations
58      IDispatch *pIDispatch=NULL;
59      HRESULT hr;
60      DISPID dispid;
61      OLECHAR *szFunctionName1 = L"SumOfTwoIntegers";
62      OLECHAR *szFunctionName2 = L"SubtractionOfTwoIntegers";
63      VARIANT vArg[2], vRet;
64      DISPPARAMS param={vArg,0,2,NULL};
65      int n1,n2;
66      TCHAR str[255];
67      // code
68      switch(iMsg)
69      {
70      case WM_CREATE:
71          // initialize COM library
72          hr=CoInitialize(NULL);
73          if(FAILED(hr))
74          {
75              ComErrorDescriptionString(hwnd, hr);
76              MessageBox(hwnd, TEXT("COM library can not be initialized"),
                  TEXT("COM Error"), MB_OK);
77              DestroyWindow(hwnd);
78              exit(0);
79          }
80          // get ISum Interface
81          hr=CoCreateInstance(CLSID_MyMath,
82                              NULL,
83                              CLSCTX_INPROC_SERVER,
84                              IID_IDispatch,
85                              (void **)&pIDispatch);
86          if(FAILED(hr))
87          {
88              ComErrorDescriptionString(hwnd, hr);
89              MessageBox(hwnd, TEXT("Component Can Not Be Created"), TEXT("COM
                    Error"), MB_OK | MB_ICONERROR | MB_TOPMOST);
90              DestroyWindow(hwnd);
91              exit(0);
92          }
93
94          // *** common code for both IMyMath->SumOfTwoIntegers() and IMyMath-
              >SubtractionOfTwoIntegers() ***
95          n1=75;
96          n2=25;
97          // as DISPPARAMS rgvarg member receives parameters in reverse order
98          VariantInit(vArg);
99          vArg[0].vt=VT_INT;
100         vArg[0].intVal=n2;
101         vArg[1].vt=VT_INT;
102         vArg[1].intVal=n1;
103         param.cArgs=2;
```

```
104         param.cNamedArgs=0;
105         param.rgdispidNamedArgs=NULL;
106         // reverse order of parameters
107         param.rgvarg=vArg;
108         // return value
109         VariantInit(&vRet);
110
111         // *** code for IMyMath->SumOfTwoIntegers() ***
112         hr = pIDispatch->GetIDsOfNames(IID_NULL,
113             &szFunctionName1,
114             1,
115             GetUserDefaultLCID(),
116             &dispid);
117         if (FAILED(hr))
118         {
119             ComErrorDescriptionString(hwnd, hr);
120             MessageBox(NULL, TEXT("Can Not Get ID For SumOfTwoIntegers()"),
                  TEXT("Error"), MB_OK | MB_ICONERROR | MB_TOPMOST);
121             pIDispatch->Release();
122             DestroyWindow(hwnd);
123         }
124         hr=pIDispatch->Invoke(dispid,
125                               IID_NULL,
126                               GetUserDefaultLCID(),
127                               DISPATCH_METHOD,
128                               &param,
129                               &vRet,
130                               NULL,
131                               NULL);
132         if(FAILED(hr))
133         {
134             ComErrorDescriptionString(hwnd, hr);
135             MessageBox(NULL, TEXT("Can Not Invoke Function"), TEXT("Error"),
                    MB_OK | MB_ICONERROR | MB_TOPMOST);
136             pIDispatch->Release();
137             DestroyWindow(hwnd);
138         }
139         else
140         {
141             wsprintf(str, TEXT("Sum Of %d And %d Is %d"), n1, n2,
                  vRet.lVal);
142             MessageBox(hwnd, str, TEXT("SumOfTwoIntegers"), MB_OK);
143         }
144
145         // *** code for IMyMath->SubtractionOfTwoIntegers() ***
146         hr = pIDispatch->GetIDsOfNames(IID_NULL,
147             &szFunctionName2,
148             1,
149             GetUserDefaultLCID(),
150             &dispid);
151         if (FAILED(hr))
152         {
153             ComErrorDescriptionString(hwnd, hr);
```

```
154             MessageBox(NULL, TEXT("Can Not Get ID For SubtractionOfTwoIntegers()"), TEXT("Error"), MB_OK | MB_ICONERROR | MB_TOPMOST);
155             pIDispatch->Release();
156             DestroyWindow(hwnd);
157         }
158         // Invoke() for IMyMath->SubtractionOfTwoIntegers()
159         hr = pIDispatch->Invoke(dispid,
160             IID_NULL,
161             GetUserDefaultLCID(),
162             DISPATCH_METHOD,
163             &param,
164             &vRet,
165             NULL,
166             NULL);
167         if (FAILED(hr))
168         {
169             ComErrorDescriptionString(hwnd, hr);
170             MessageBox(NULL, TEXT("Can Not Invoke Function"), TEXT("Error"), MB_OK | MB_ICONERROR | MB_TOPMOST);
171             pIDispatch->Release();
172             DestroyWindow(hwnd);
173         }
174         else
175         {
176             wsprintf(str, TEXT("Subtraction Of %d And %d Is %d"), n1, n2, vRet.lVal);
177             MessageBox(hwnd, str, TEXT("SubtractionOfTwoIntegers"), MB_OK);
178         }
179
180         // clean-up
181         VariantClear(vArg);
182         VariantClear(&vRet);
183         pIDispatch->Release();
184         pIDispatch = NULL;
185         DestroyWindow(hwnd);
186         break;
187     case WM_DESTROY:
188         CoUninitialize();
189         PostQuitMessage(0);
190         break;
191     }
192     return(DefWindowProc(hwnd,iMsg,wParam,lParam));
193 }
194
195 void ComErrorDescriptionString(HWND hwnd, HRESULT hr)
196 {
197     // variable declarations
198     TCHAR* szErrorMessage = NULL;
199     TCHAR str[255];
200     // code
201     if (FACILITY_WINDOWS == HRESULT_FACILITY(hr))
202         hr = HRESULT_CODE(hr);
```

```
203
204     if (FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |                    ↵
          FORMAT_MESSAGE_FROM_SYSTEM, NULL, hr, MAKELANGID(LANG_NEUTRAL,       ↵
          SUBLANG_DEFAULT), (LPTSTR)&szErrorMessage, 0, NULL) != 0)
205     {
206         swprintf_s(str, TEXT("%s"), szErrorMessage);
207         LocalFree(szErrorMessage);
208     }
209     else
210         swprintf_s(str, TEXT("[Could not find a description for error # %  ↵
            #x.]\n"), hr);
211
212     MessageBox(hwnd, str, TEXT("COM Error"), MB_OK);
213 }
214
```