```cpp
#include<windows.h>
#include <stdio.h> // for swprintf_s()
#include"AutomationServerWithRegFile.h"
// global function declarations
LRESULT CALLBACK WndProc(HWND,UINT,WPARAM,LPARAM);
// global variable declarations
IMyMath *pIMyMath=NULL;
// WinMain
int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine,int nCmdShow)
{
    // variable declarations
    WNDCLASSEX wndclass;
    HWND hwnd;
    MSG msg;
    TCHAR AppName[]=TEXT("ComClient");
    HRESULT hr;
    // code
    // COM Initialization
    hr=CoInitialize(NULL);
    if(FAILED(hr))
    {
        MessageBox(NULL,TEXT("COM Library Can Not Be Initialized.\nProgram 
          Will Now Exit."),TEXT("Program Error"),MB_OK);
        exit(0);
    }
    // WNDCLASSEX initialization
    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 COM Dll 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);
    }
    // COM Un-initialization
    CoUninitialize();
    return((int)msg.wParam);
}
// Window Procedure
LRESULT CALLBACK WndProc(HWND hwnd,UINT iMsg,WPARAM wParam,LPARAM lParam)
{
    // function declarations
    void ComErrorDescriptionString(HWND, HRESULT);
    void SafeInterfaceRelease(void);
    // variable declarations
    HRESULT hr;
    int iNum1,iNum2,iSum,iSubtract;
    TCHAR str[255];
    // code
    switch(iMsg)
    {
    case WM_CREATE:
        hr=CoCreateInstance(CLSID_MyMath,NULL,CLSCTX_INPROC_SERVER,
                            IID_IMyMath,(void **)&pIMyMath);
        if(FAILED(hr))
        {
            ComErrorDescriptionString(hwnd, hr);
            DestroyWindow(hwnd);
        }
        // initialize arguments hardcoded
        iNum1=155;
        iNum2=145;
        // call SumOfTwoIntegers() of IMyMath to get the sum
        pIMyMath->SumOfTwoIntegers(iNum1,iNum2,&iSum);
        wsprintf(str, TEXT("Sum Of %d And %d Is %d"), iNum1, iNum2, iSum);
        MessageBox(hwnd, str, TEXT("SumOfTwoIntegers"), MB_OK);

        // call SumOfTwoIntegers() of IMyMath to get the sum
        pIMyMath->SubtractionOfTwoIntegers(iNum1, iNum2, &iSubtract);
        wsprintf(str, TEXT("Subtraction Of %d And %d Is %d"), iNum1, iNum2, 
          iSubtract);
        MessageBox(hwnd, str, TEXT("SubtractionOfTwoIntegers"), MB_OK);

        // release
        pIMyMath->Release();
        pIMyMath=NULL;// make relesed interface NULL
        // exit the application
        DestroyWindow(hwnd);
        break;
```

```cpp
    case WM_DESTROY:
        SafeInterfaceRelease();
        PostQuitMessage(0);
        break;
    }
    return(DefWindowProc(hwnd,iMsg,wParam,lParam));
}
void ComErrorDescriptionString(HWND hwnd, HRESULT hr)
{
    // variable declarations
    TCHAR* szErrorMessage = NULL;
    TCHAR str[255];
    // code
    if (FACILITY_WINDOWS == HRESULT_FACILITY(hr))
        hr = HRESULT_CODE(hr);

    if (FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM, NULL, hr, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), (LPTSTR)&szErrorMessage, 0, NULL) != 0)
    {
        swprintf_s(str, TEXT("%s"), szErrorMessage);
        LocalFree(szErrorMessage);
    }
    else
        swprintf_s(str, TEXT("[Could not find a description for error # %#x.]\n"), hr);

    MessageBox(hwnd, str, TEXT("COM Error"), MB_OK);
}
void SafeInterfaceRelease(void)
{
    // code
    if(pIMyMath)
    {
        pIMyMath->Release();
        pIMyMath=NULL;
    }
}
```