```
#define UNICODE
#include<windows.h>
#include"HeaderForClientOfAggregationComponentWithRegFile.h"
// global function declarations
LRESULT CALLBACK WndProc(HWND,UINT,WPARAM,LPARAM);
// global variable declarations
ISum *pISum=NULL;
ISubtract *pISubtract=NULL;
IMultiplication *pIMultiplication=NULL;
IDivision *pIDivision=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,
```

```cpp
                      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 SafeInterfaceRelease(void);
    // variable declarations
    HRESULT hr;
    int iNum1,iNum2,iSum,iSubtraction,iMultiplication,iDivision;
    TCHAR str[255];
    // code
    switch(iMsg)
    {
    case WM_CREATE:
        hr=CoCreateInstance(CLSID_SumSubtract,NULL,CLSCTX_INPROC_SERVER,
                            IID_ISum,(void **)&pISum);
        if(FAILED(hr))
        {
            MessageBox(hwnd,TEXT("ISum Interface Can Not Be Obtained"),TEXT("Error"),MB_OK);
            DestroyWindow(hwnd);
        }
        // initialize arguments hardcoded
        iNum1=65;
        iNum2=45;
        // call SumOfTwoIntegers() of ISum to get the sum
        pISum->SumOfTwoIntegers(iNum1,iNum2,&iSum);
        // display the result
        wsprintf(str,TEXT("Sum Of %d And %d = %d"),iNum1,iNum2,iSum);
        MessageBox(hwnd,str,TEXT("Result"),MB_OK);
        // call QueryInterface() on ISum,to get ISubtract's pointer
        hr=pISum->QueryInterface(IID_ISubtract,(void **)&pISubtract);
        if(FAILED(hr))
        {
            MessageBox(hwnd,TEXT("ISubtract Interface Can Not Be Obtained"),TEXT("Error"),MB_OK);
            DestroyWindow(hwnd);
        }
        // as ISum is now not needed onwords, release it
        pISum->Release();
        pISum=NULL;// make relesed interface NULL
        // again initialize arguments hardcoded
        iNum1=155;
        iNum2=55;
        // call SubtractionOfTwoIntegers() of ISubtract to get the subtraction
        pISubtract->SubtractionOfTwoIntegers(iNum1,iNum2,&iSubtraction);
        // display the result
        wsprintf(str,TEXT("Subtraction Of %d And %d = %d"),iNum1,iNum2,iSubtraction);
        MessageBox(hwnd,str,TEXT("Result"),MB_OK);
        // call QueryInterface() on ISubtract,to get IMultiplication's pointer
        hr=pISubtract->QueryInterface(IID_IMultiplication,(void **)&pIMultiplication);
        if(FAILED(hr))
        {
            MessageBox(hwnd,TEXT("IMultiplication Interface Can Not Be Obtained"),TEXT("Error"),MB_OK);
            DestroyWindow(hwnd);
        }
        // as ISubtract is now not needed onwords, release it
        pISubtract->Release();
        pISubtract=NULL;// make relesed interface NULL
        // again initialize arguments hardcoded
        iNum1=30;
        iNum2=25;
        // call MultiplicationOfTwoIntegers() of IMultiplication to get the Multiplication
        pIMultiplication->MultiplicationOfTwoIntegers(iNum1,iNum2,&iMultiplication);
        // display the result
        wsprintf(str,TEXT("Multiplication Of %d And %d = %d"),iNum1,iNum2,iMultiplication);
        MessageBox(hwnd,str,TEXT("Result"),MB_OK);
        // call QueryInterface() on IMultiplication's to get IDivision pointer
        hr=pIMultiplication->QueryInterface(IID_IDivision,(void **)&pIDivision);
        if(FAILED(hr))
        {
            MessageBox(hwnd,TEXT("IDivision Interface Can Not Be Obtained"),TEXT("Error"),MB_OK);
            DestroyWindow(hwnd);
        }
        // as IMultiplication is now not needed onwords, release it
        pIMultiplication->Release();
        pIMultiplication=NULL;// make relesed interface NULL
        // again initialize arguments hardcoded
        iNum1=200;
        iNum2=25;
        // call DivisionOfTwoIntegers() of IDivision to get the Division
        pIDivision->DivisionOfTwoIntegers(iNum1,iNum2,&iDivision);
        // display the result
        wsprintf(str,TEXT("Division Of %d And %d = %d"),iNum1,iNum2,iDivision);
        MessageBox(hwnd,str,TEXT("Result"),MB_OK);
        // finally release IDivision
        pIDivision->Release();
        pIDivision=NULL;// make relesed interface NULL
        // exit the application
        DestroyWindow(hwnd);
        break;
    case WM_DESTROY:
        SafeInterfaceRelease();
        PostQuitMessage(0);
        break;
    }
    return(DefWindowProc(hwnd,iMsg,wParam,lParam));
}
void SafeInterfaceRelease(void)
{
    // code
    if(pISum)
    {
        pISum->Release();
        pISum=NULL;
    }
    if(pISubtract)
    {
        pISubtract->Release();
        pISubtract=NULL;
    }
    if(pIMultiplication)
    {
        pIMultiplication->Release();
        pIMultiplication=NULL;
    }
    if(pIDivision)
    {
        pIDivision->Release();
        pIDivision=NULL;
    }
}

```