

```

1  #define UNICODE
2  #include<windows.h>
3  #include<process.h>
4  #include"HeaderForClientOfContainmentComponentWithRegFile.h"
5  // global function declarations
6  LRESULT CALLBACK WndProc(HWND,UINT,WPARAM,LPARAM);
7  // global variable declarations
8  ISum *pISum=NULL;
9  ISubtract *pISubtract=NULL;
10 IMultiplication *pIMultiplication=NULL;
11 IDivision *pIDivision=NULL;
12 // WinMain
13 int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,
14 LPSTR lpCmdLine,int nCmdShow)
15 {
16     // variable declarations
17     WNDCLASSEX wndclass;
18     HWND hwnd;
19     MSG msg;
20     TCHAR AppName[]=TEXT("ComClient");
21     HRESULT hr;
22     // code
23     // COM Initialization
24     hr=CoInitialize(NULL);
25     if(FAILED(hr))
26     {
27         MessageBox(NULL,TEXT("COM Library Can Not Be Initialized.\nProgram Will
28         Now Exit."),TEXT("Program Error"),MB_OK);
29         exit(0);
30     }
31     // WNDCLASSEX initialization
32     wndclass.cbSize=sizeof(wndclass);
33     wndclass.style=CS_HREDRAW|CS_VREDRAW;
34     wndclass.cbClsExtra=0;
35     wndclass.cbWndExtra=0;
36     wndclass.lpfNWndProc=WndProc;
37     wndclass.hIcon=LoadIcon(NULL,IDI_APPLICATION);
38     wndclass.hCursor=LoadCursor(NULL,IDC_ARROW);
39     wndclass.hbrBackground=(HBRUSH)GetStockObject(WHITE_BRUSH);
40     wndclass.hInstance=hInstance;
41     wndclass.lpszClassName=AppName;
42     wndclass.lpszMenuName=NULL;
43     wndclass.hIconSm=LoadIcon(NULL,IDI_APPLICATION);
44     // register window class
45     RegisterClassEx(&wndclass);
46     // create window
47     hwnd=CreateWindow(AppName,
48         TEXT("Client Of COM Dll Server"),
49         WS_OVERLAPPEDWINDOW,
50         CW_USEDEFAULT,
51         CW_USEDEFAULT,
52         CW_USEDEFAULT,
53         CW_USEDEFAULT,
54         NULL,
55         NULL,
56         hInstance,
57         NULL);
58     if(hwnd==NULL)
59     {
60         MessageBox(NULL,TEXT("Can Not Create Window"),TEXT("Error"),MB_OK);
61         exit(0);
62     }
63     ShowWindow(hwnd,nCmdShow);
64     UpdateWindow(hwnd);
65     // Main Message Loop
66     while(msg.message!=WM_QUIT)
67     {
68         if(!GetMessage(&msg,hwnd,0,0))
69             break;
70         TranslateMessage(&msg);
71         DispatchMessage(&msg);
72     }
73     return msg.wParam;
74 }

```

```

52             CW_USEDEFAULT,
53             NULL,
54             NULL,
55             hInstance,
56             NULL);
57     ShowWindow(hwnd,nCmdShow);
58     UpdateWindow(hwnd);
59     // message loop
60     while(GetMessage(&msg,NULL,0,0))
61     {
62         TranslateMessage(&msg);
63         DispatchMessage(&msg);
64     }
65     // COM Un-initialization
66     CoUninitialize();
67     return((int)msg.wParam);
68 }
69 // Window Procedure
70 LRESULT CALLBACK WndProc(HWND hwnd,UINT iMsg,WPARAM wParam,LPARAM lParam)
71 {
72     // function declarations
73     void SafeInterfaceRelease(void);
74     // variable declarations
75     HRESULT hr;
76     int iNum1,iNum2,iSum,iSubtraction,iMultiplication,iDivision;
77     TCHAR str[255];
78     // code
79     switch(iMsg)
80     {
81     case WM_CREATE:
82         hr=CoCreateInstance(CLSID_SumSubtract,NULL,CLSCTX_INPROC_SERVER,
83             IID_ISum,(void **)&pISum);
84         if(FAILED(hr))
85         {
86             MessageBox(hwnd,TEXT("ISum Interface Can Not Be Obtained"),TEXT
87                 ("Error"),MB_OK);
88             DestroyWindow(hwnd);
89         }
90         // initialize arguments hardcoded
91         iNum1=65;
92         iNum2=45;
93         // call SumOfTwoIntegers() of ISum to get the sum
94         pISum->SumOfTwoIntegers(iNum1,iNum2,&iSum);
95         // display the result
96         wsprintf(str,TEXT("Sum Of %d And %d = %d"),iNum1,iNum2,iSum);
97         MessageBox(hwnd,str,TEXT("Result"),MB_OK);
98         // call QueryInterface() on ISum,to get ISubtract's pointer
99         hr=pISum->QueryInterface(IID_ISubtract,(void **)&pISubtract);
100         if(FAILED(hr))
101         {
102             MessageBox(hwnd,TEXT("ISubtract Interface Can Not Be Obtained"),TEXT
103                 ("Error"),MB_OK);

```

```

102         DestroyWindow(hwnd);
103     }
104     // as ISum is now not needed onwords, release it
105     pISum->Release();
106     pISum=NULL;// make relese interface NULL
107     // again initialize arguments hardcoded
108     iNum1=155;
109     iNum2=55;
110     // call SubtractionOfTwoIntegers() of ISubtract to get the subtraction
111     pISubtract->SubtractionOfTwoIntegers(iNum1,iNum2,&iSubtraction);
112     // display the result
113     wsprintf(str,TEXT("Subtraction Of %d And %d = %d"),iNum1,iNum2,iSubtraction);
114     MessageBox(hwnd,str,TEXT("Result"),MB_OK);
115     // call QueryInterface() on ISubtract,to get IMultiplication's pointer
116     hr=pISubtract->QueryInterface(IID_IMultiplication,(void **)&pIMultiplication);
117     if(FAILED(hr))
118     {
119         MessageBox(hwnd,TEXT("IMultiplication Interface Can Not Be
120         Obtained"),TEXT("Error"),MB_OK);
121         DestroyWindow(hwnd);
122     }
123     // as ISubtract is now not needed onwords, release it
124     pISubtract->Release();
125     pISubtract=NULL;// make relese interface NULL
126     // again initialize arguments hardcoded
127     iNum1=30;
128     iNum2=25;
129     // call MultiplicationOfTwoIntegers() of IMultiplication to get the
130     Multiplication
131     pIMultiplication->MultiplicationOfTwoIntegers
132     (iNum1,iNum2,&iMultiplication);
133     // display the result
134     wsprintf(str,TEXT("Multiplication Of %d And %d = %d"),iNum1,iNum2,iMultiplication);
135     MessageBox(hwnd,str,TEXT("Result"),MB_OK);
136     // call QueryInterface() on IMultiplication's to get IDivision pointer
137     hr=pIMultiplication->QueryInterface(IID_IDivision,(void **)&pIDivision);
138     if(FAILED(hr))
139     {
140         MessageBox(hwnd,TEXT("IDivision Interface Can Not Be Obtained"),TEXT
141         ("Error"),MB_OK);
142         DestroyWindow(hwnd);
143     }
144     // as IMultiplication is now not needed onwords, release it
145     pIMultiplication->Release();
146     pIMultiplication=NULL;// make relese interface NULL
147     // again initialize arguments hardcoded
148     iNum1=200;
149     iNum2=25;
150     // call DivisionOfTwoIntegers() of IDivision to get the Division

```

```
147     pIDivision->DivisionOfTwoIntegers(iNum1,iNum2,&iDivision);
148     // display the result
149     wsprintf(str,TEXT("Division Of %d And %d = %d"),iNum1,iNum2,iDivision);
150     MessageBox(hwnd,str,TEXT("Result"),MB_OK);
151     // finally release IDivision
152     pIDivision->Release();
153     pIDivision=NULL;// make relesead interface NULL
154     // exit the application
155     DestroyWindow(hwnd);
156     break;
157 case WM_DESTROY:
158     SafeInterfaceRelease();
159     PostQuitMessage(0);
160     break;
161 }
162 return(DefWindowProc(hwnd,iMsg,wParam,lParam));
163 }
164 void SafeInterfaceRelease(void)
165 {
166     // code
167     if(pISum)
168     {
169         pISum->Release();
170         pISum=NULL;
171     }
172     if(pISubtract)
173     {
174         pISubtract->Release();
175         pISubtract=NULL;
176     }
177     if(pIMultiplication)
178     {
179         pIMultiplication->Release();
180         pIMultiplication=NULL;
181     }
182     if(pIDivision)
183     {
184         pIDivision->Release();
185         pIDivision=NULL;
186     }
187 }
188 }
```