```
#define UNICODE
#include<windows.h>
#include"ContainmentInnerComponentWithRegFile.h"
#include"ContainmentOuterComponentWithRegFile.h"
// class declarations
class CSumSubtract:public ISum,ISubtract,IMultiplication,IDivision
{
private:
    long m_cRef;
    IMultiplication *m_pIMultiplication;
    IDivision *m_pIDivision;
public:
    // constructor method declarations
    CSumSubtract(void);
    // destructor method declarations
    ~CSumSubtract(void);
    // IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface(REFIID,void **);
    ULONG __stdcall AddRef(void);
    ULONG __stdcall Release(void);
    // ISum specific method declarations (inherited)
    HRESULT __stdcall SumOfTwoIntegers(int,int,int *);
    // ISubtract specific method declarations (inherited)
    HRESULT __stdcall SubtractionOfTwoIntegers(int,int,int *);
    // IMultiplication specific method declarations (inherited)
    HRESULT __stdcall MultiplicationOfTwoIntegers(int,int,int *);
    // IDivision specific method declarations (inherited)
    HRESULT __stdcall DivisionOfTwoIntegers(int,int,int *);
    // custom method for inner component creation
    HRESULT __stdcall InitializeInnerComponent(void);
};
class CSumSubtractClassFactory:public IClassFactory
{
private:
    long m_cRef;
public:
    // constructor method declarations
    CSumSubtractClassFactory(void);
    // destructor method declarations
    ~CSumSubtractClassFactory(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);
};
// global variable declarations
long glNumberOfActiveComponents=0;// number of active components
long glNumberOfServerLocks=0;// number of locks on this dll
// DllMain
```

```
BOOL WINAPI DllMain(HINSTANCE hDll,DWORD dwReason,LPVOID Reserved)
{
    // code
    switch(dwReason)
    {
    case DLL_PROCESS_ATTACH:
        break;
    case DLL_PROCESS_DETACH:
        break;
    }
return(TRUE);
}
// Implementation Of CSumSubtract's Constructor Method
CSumSubtract::CSumSubtract(void)
{
    // code
    // initialization of private data members
    m_pIMultiplication=NULL;
    m_pIDivision=NULL;
    m_cRef=1;// hardcoded initialization to anticipate possible failure of QueryInterface()
    InterlockedIncrement(&glNumberOfActiveComponents);// increment global counter
}
// Implementation Of CSumSubtract's Destructor Method
CSumSubtract::~CSumSubtract(void)
{
    // code
    InterlockedDecrement(&glNumberOfActiveComponents);// decrement global counter
    if(m_pIMultiplication)
    {
        m_pIMultiplication->Release();
        m_pIMultiplication=NULL;
    }
    if(m_pIDivision)
    {
        m_pIDivision->Release();
        m_pIDivision=NULL;
    }
}
// Implementation Of CSumSubtract's IUnknown's Methods
HRESULT CSumSubtract::QueryInterface(REFIID riid,void **ppv)
{
    // code
    if(riid==IID_IUnknown)
        *ppv=static_cast<ISum *>(this);
    else if(riid==IID_ISum)
        *ppv=static_cast<ISum *>(this);
    else if(riid==IID_ISubtract)
        *ppv=static_cast<ISubtract *>(this);
    else if(riid==IID_IMultiplication)
        *ppv=static_cast<IMultiplication *>(this);
    else if(riid==IID_IDivision)
```

```
        *ppv=static_cast<IDivision *>(this);
    else
    {
        *ppv=NULL;
        return(E_NOINTERFACE);
    }
    reinterpret_cast<IUnknown *>(*ppv)->AddRef();
    return(S_OK);
}
ULONG CSumSubtract::AddRef(void)
{
    // code
    InterlockedIncrement(&m_cRef);
    return(m_cRef);
}
ULONG CSumSubtract::Release(void)
{
    // code
    InterlockedDecrement(&m_cRef);
    if(m_cRef==0)
    {
        delete(this);
        return(0);
    }
    return(m_cRef);
}
// Implementation Of ISum's Methods
HRESULT CSumSubtract::SumOfTwoIntegers(int num1,int num2,int *pSum)
{
    // code
    *pSum=num1+num2;
    return(S_OK);
}
// Implementation Of ISubtract's Methods
HRESULT CSumSubtract::SubtractionOfTwoIntegers(int num1,int num2,int *pSubtract)
{
    // code
    *pSubtract=num1-num2;
    return(S_OK);
}
// Implementation Of IMultiplication's Methods
HRESULT CSumSubtract::MultiplicationOfTwoIntegers(int num1,int num2,int
  *pMultiplication)
{
    // code
    // delegate to inner component
    m_pIMultiplication->MultiplicationOfTwoIntegers(num1,num2,pMultiplication);
    return(S_OK);
}
// Implementation Of IDivision's Methods
HRESULT CSumSubtract::DivisionOfTwoIntegers(int num1,int num2,int *pDivision)
{
```

```cpp
    // code
    // delegate to inner component
    m_pIDivision->DivisionOfTwoIntegers(num1,num2,pDivision);
    return(S_OK);
}
HRESULT CSumSubtract::InitializeInnerComponent(void)
{
    // variable declarations
    HRESULT hr;
    // code
    hr=CoCreateInstance(CLSID_MultiplicationDivision,NULL,CLSCTX_INPROC_SERVER,
                        IID_IMultiplication,(void **)&m_pIMultiplication);
    if(FAILED(hr))
    {
        MessageBox(NULL,TEXT("IMultiplication Interface Can Not Be Obtained From Inner Component."),TEXT("Error"),MB_OK);
        return(E_FAIL);
    }
    hr=m_pIMultiplication->QueryInterface(IID_IDivision,(void **)&m_pIDivision);
    if(FAILED(hr))
    {
        MessageBox(NULL,TEXT("IDivision Interface Can Not Be Obtained From Inner Component."),TEXT("Error"),MB_OK);
        return(E_FAIL);
    }
    return(S_OK);
}
// Implementation Of CSumSubtractClassFactory's Constructor Method
CSumSubtractClassFactory::CSumSubtractClassFactory(void)
{
    // code
    m_cRef=1;// hardcoded initialization to anticipate possible failure of QueryInterface()
}
// Implementation Of CSumSubtractClassFactory's Destructor Method
CSumSubtractClassFactory::~CSumSubtractClassFactory(void)
{
    // code
}
// Implementation Of CSumSubtractClassFactory's IClassFactory's IUnknown's Methods
HRESULT CSumSubtractClassFactory::QueryInterface(REFIID riid,void **ppv)
{
    // code
    if(riid==IID_IUnknown)
        *ppv=static_cast<IClassFactory *>(this);
    else if(riid==IID_IClassFactory)
        *ppv=static_cast<IClassFactory *>(this);
    else
    {
        *ppv=NULL;
        return(E_NOINTERFACE);
    }
    reinterpret_cast<IUnknown *>(*ppv)->AddRef();
    return(S_OK);
}
ULONG CSumSubtractClassFactory::AddRef(void)
{
    // code
    InterlockedIncrement(&m_cRef);
    return(m_cRef);
}
ULONG CSumSubtractClassFactory::Release(void)
{
    // code
    InterlockedDecrement(&m_cRef);
    if(m_cRef==0)
    {
        delete(this);
        return(0);
    }
    return(m_cRef);
}
// Implementation Of CSumSubtractClassFactory's IClassFactory's Methods
HRESULT CSumSubtractClassFactory::CreateInstance(IUnknown *pUnkOuter,REFIID
  riid,void **ppv)
{
    // variable declarations
    CSumSubtract *pCSumSubtract=NULL;
    HRESULT hr;
    // code
    if(pUnkOuter!=NULL)
        return(CLASS_E_NOAGGREGATION);
    // create the instance of component i.e. of CSumSubtract class
    pCSumSubtract=new CSumSubtract;
    if(pCSumSubtract==NULL)
        return(E_OUTOFMEMORY);
    // initialize the inner component
    hr=pCSumSubtract->InitializeInnerComponent();
    if(FAILED(hr))
    {
        MessageBox(NULL,TEXT("Failed To Initialize Inner Component"),TEXT
          ("Error"),MB_OK);
        pCSumSubtract->Release();
        return(hr);
    }
    // get the requested interface
    hr=pCSumSubtract->QueryInterface(riid,ppv);
    pCSumSubtract->Release();// anticipate possible failure of QueryInterface()
    return(hr);
}
HRESULT CSumSubtractClassFactory::LockServer(BOOL fLock)
{
    // code
    if(fLock)
        InterlockedIncrement(&glNumberOfServerLocks);
    else
        InterlockedDecrement(&glNumberOfServerLocks);
    return(S_OK);
}
// Implementation Of Exported Functions From This Dll
HRESULT __stdcall DllGetClassObject(REFCLSID rclsid,REFIID riid,void **ppv)
{
    // variable declaraions
    CSumSubtractClassFactory *pCSumSubtractClassFactory=NULL;
    HRESULT hr;
    // code
    if(rclsid!=CLSID_SumSubtract)
        return(CLASS_E_CLASSNOTAVAILABLE);
    // create class factory
    pCSumSubtractClassFactory=new CSumSubtractClassFactory;
    if(pCSumSubtractClassFactory==NULL)
        return(E_OUTOFMEMORY);
    hr=pCSumSubtractClassFactory->QueryInterface(riid,ppv);
    pCSumSubtractClassFactory->Release();// anticipate possible failure of
      QueryInterface()
    return(hr);
}
HRESULT __stdcall DllCanUnloadNow(void)
{
    // code
    if((glNumberOfActiveComponents==0) && (glNumberOfServerLocks==0))
        return(S_OK);
    else
        return(S_FALSE);
}

```