```
#define UNICODE
#include<windows.h>
#include"AggregationInnerComponentWithRegFile.h"
// interface declaration ( for internal use only. i.e. not to be included in .h file )
interface INoAggregationIUnknown // new
{
    virtual HRESULT __stdcall QueryInterface_NoAggregation(REFIID,void **)=0;
    virtual ULONG __stdcall AddRef_NoAggregation(void)=0;
    virtual ULONG __stdcall Release_NoAggregation(void)=0;
};
// class declarations
class CMultiplicationDivision:public INoAggregationIUnknown,IMultiplication,IDivision
{
private:
    long m_cRef;
    IUnknown *m_pIUnknownOuter;
public:
    // constructor method declarations
    CMultiplicationDivision(IUnknown *);// new
    // destructor method declarations
    ~CMultiplicationDivision(void);
    // Aggregation Supported IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface(REFIID,void **);
    ULONG __stdcall AddRef(void);
    ULONG __stdcall Release(void);
    // Aggregation NonSupported IUnknown specific method declarations (inherited)
    HRESULT __stdcall QueryInterface_NoAggregation(REFIID,void **);// new
    ULONG __stdcall AddRef_NoAggregation(void);// new
    ULONG __stdcall Release_NoAggregation(void);// new
    // IMultiplication specific method declarations (inherited)
    HRESULT __stdcall MultiplicationOfTwoIntegers(int,int,int *);
    // IDivision specific method declarations (inherited)
    HRESULT __stdcall DivisionOfTwoIntegers(int,int,int *);
};
class CMultiplicationDivisionClassFactory:public IClassFactory
{
private:
    long m_cRef;
public:
    // constructor method declarations
    CMultiplicationDivisionClassFactory(void);
    // destructor method declarations
    ~CMultiplicationDivisionClassFactory(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);
```

```cpp
};
// 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 CMultiplicationDivision's Constructor Method
CMultiplicationDivision::CMultiplicationDivision(IUnknown *pIUnknownOuter)
{
    // code
    m_cRef=1;// hardcoded initialization to anticipate possible failure of 
      QueryInterface()
    InterlockedIncrement(&glNumberOfActiveComponents);// increment global counter
    if(pIUnknownOuter!=NULL)
        m_pIUnknownOuter=pIUnknownOuter;
    else
        m_pIUnknownOuter=reinterpret_cast<IUnknown *>
          (static_cast<INoAggregationIUnknown *>(this));
}
// Implementation Of CSumSubtract's Destructor Method
CMultiplicationDivision::~CMultiplicationDivision(void)
{
    // code
    InterlockedDecrement(&glNumberOfActiveComponents);// decrement global counter
}
// Implementation Of CMultiplicationDivision's Aggrgation Supporting IUnknown's 
  Methods
HRESULT CMultiplicationDivision::QueryInterface(REFIID riid,void **ppv)
{
    // code
    return(m_pIUnknownOuter->QueryInterface(riid,ppv));
}
ULONG CMultiplicationDivision::AddRef(void)
{
    // code
    return(m_pIUnknownOuter->AddRef());
}
ULONG CMultiplicationDivision::Release(void)
{
    // code
    return(m_pIUnknownOuter->Release());
```

```
}
// Implementation Of CMultiplicationDivision's Aggregation NonSupporting IUnknown's Methods
HRESULT CMultiplicationDivision::QueryInterface_NoAggregation(REFIID riid,void **ppv)
{
    // code
    if(riid==IID_IUnknown)
        *ppv=static_cast<INoAggregationIUnknown *>(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 CMultiplicationDivision::AddRef_NoAggregation(void)
{
    // code
    InterlockedIncrement(&m_cRef);
    return(m_cRef);
}
ULONG CMultiplicationDivision::Release_NoAggregation(void)
{
    // code
    InterlockedDecrement(&m_cRef);
    if(m_cRef==0)
    {
        delete(this);
        return(0);
    }
    return(m_cRef);
}
// Implementation Of IMultiplication's Methods
HRESULT CMultiplicationDivision::MultiplicationOfTwoIntegers(int num1,int num2,int *pMultiplication)
{
    // code
    *pMultiplication=num1*num2;
    return(S_OK);
}
// Implementation Of IDivision's Methods
HRESULT CMultiplicationDivision::DivisionOfTwoIntegers(int num1,int num2,int *pDivision)
{
    // code
    *pDivision=num1/num2;
    return(S_OK);
}
// Implementation Of CMultiplicationDivisionClassFactory's Constructor Method
CMultiplicationDivisionClassFactory::CMultiplicationDivisionClassFactory(void)
{
    // code
    m_cRef=1;// hardcoded initialization to anticipate possible failure of QueryInterface()
}
// Implementation Of CMultiplicationDivisionClassFactory's Destructor Method
CMultiplicationDivisionClassFactory::~CMultiplicationDivisionClassFactory(void)
{
    // code
}
// Implementation Of CMultiplicationDivisionClassFactory's IClassFactory's IUnknown's Methods
HRESULT CMultiplicationDivisionClassFactory::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 CMultiplicationDivisionClassFactory::AddRef(void)
{
    // code
    InterlockedIncrement(&m_cRef);
    return(m_cRef);
}
ULONG CMultiplicationDivisionClassFactory::Release(void)
{
    // code
    InterlockedDecrement(&m_cRef);
    if(m_cRef==0)
    {
        delete(this);
        return(0);
    }
    return(m_cRef);
}
// Implementation Of CMultiplicationDivisionClassFactory's IClassFactory's Methods
HRESULT CMultiplicationDivisionClassFactory::CreateInstance(IUnknown
```

```cpp
    *pUnkOuter,REFIID riid,void **ppv)
{
    // variable declarations
    CMultiplicationDivision *pCMultiplicationDivision=NULL;
    HRESULT hr;
    // code
    if((pUnkOuter!=NULL) && (riid!=IID_IUnknown))
        return(CLASS_E_NOAGGREGATION);
    // create the instance of component i.e. of CMultiplicationDivision class
    pCMultiplicationDivision=new CMultiplicationDivision(pUnkOuter);
    if(pCMultiplicationDivision==NULL)
        return(E_OUTOFMEMORY);
    // get the requested interface
    hr=pCMultiplicationDivision->QueryInterface_NoAggregation(riid,ppv);
    pCMultiplicationDivision->Release_NoAggregation();// anticipate possible failure of QueryInterface()
    return(hr);
}
HRESULT CMultiplicationDivisionClassFactory::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
    CMultiplicationDivisionClassFactory *pCMultiplicationDivisionClassFactory=NULL;
    HRESULT hr;
    // code
    if(rclsid!=CLSID_MultiplicationDivision)
        return(CLASS_E_CLASSNOTAVAILABLE);
    // create class factory
    pCMultiplicationDivisionClassFactory=new CMultiplicationDivisionClassFactory;
    if(pCMultiplicationDivisionClassFactory==NULL)
        return(E_OUTOFMEMORY);
    hr=pCMultiplicationDivisionClassFactory->QueryInterface(riid,ppv);
    pCMultiplicationDivisionClassFactory->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);
}

```