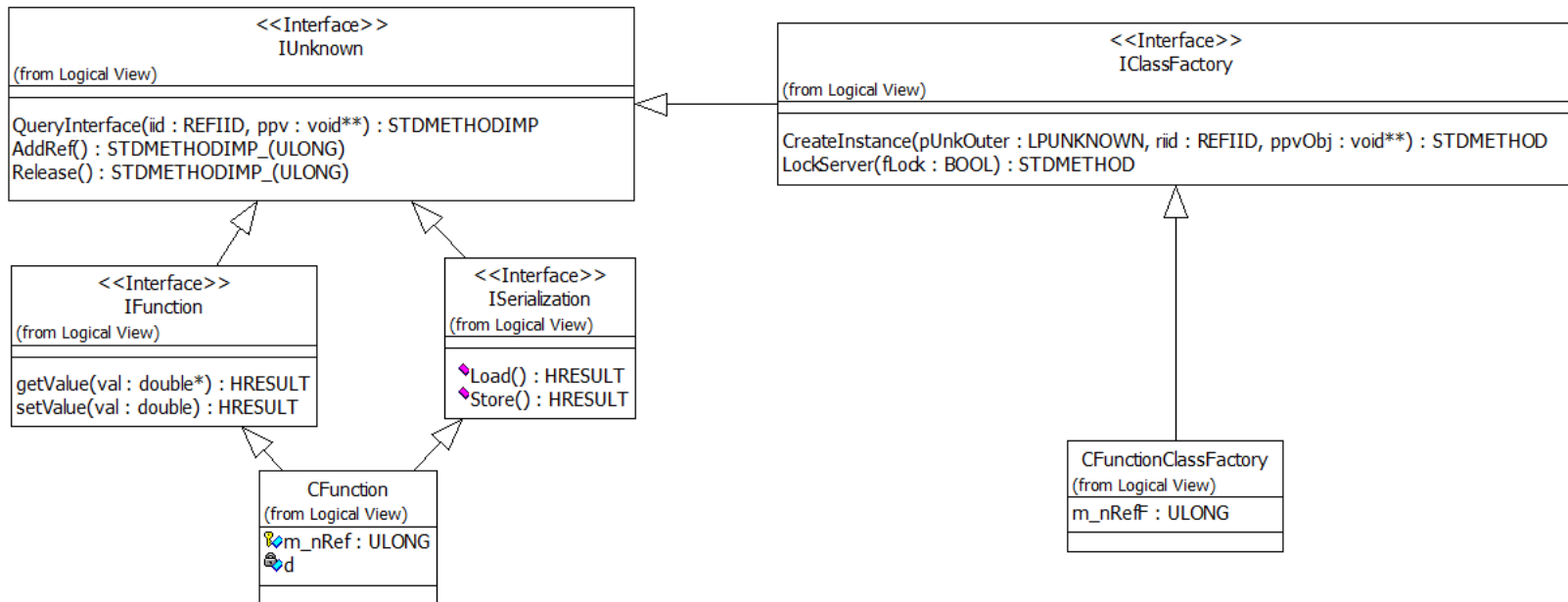


## Лекция 4: реализация COM – сервера (*in-process server*)



## Определение интерфейсов, производных от *IUnknown*

*root.h*

```
class IFunction : public IUnknown{
public:
    STDMETHOD(getValue)(double*)=0;
    STDMETHOD(setValue)(double)=0;
};
```

```
class ISerialization : public IUnknown{
public:
    STDMETHOD(Load)()=0;
    STDMETHOD(Store)()=0;
};
```

```
/****** OLE macros: *****/
```

```
#define STDMETHODCALLTYPE __stdcall
```

```
#define STDMETHOD(method) virtual HRESULT STDMETHODCALLTYPE method
```

```
#define STDMETHOD_(type,method) virtual type STDMETHODCALLTYPE method
```

```
#define STDMETHODIMP HRESULT STDMETHODCALLTYPE
```

```
#define STDMETHODIMP_(type) type STDMETHODCALLTYPE
```

```
#define SUCCEEDED(Status) ((HRESULT)(Status) >= 0)
```

```
*****/
```



## Объявление класса, наследующего интерфейсы *IFunction* и *ISerialization* (интерфейсы компонента)

*CFunction.h*

```
#include "root.h"
extern ULONG g_cLock;

class CFunction: public IFunction, ISerialization{
public:
    CFunction(){
        m_nRef=0;
    }
    // методы наследуемые от IUnknown
    STDMETHOD(QueryInterface)(REFIID, void**);
    STDMETHOD_(ULONG, AddRef)();
    STDMETHOD_(ULONG, Release)();
    // методы наследуемые от IFunction
    STDMETHOD(getValue)(double*);
    STDMETHOD(setValue)(double);
    // методы наследуемые от ISerialization
    STDMETHOD(Load)();
    STDMETHOD(Store)();
protected:
    ULONG m_nRef;
private:
    double d;
};
```



## Объявление класса, реализующего фабрику классов, наследующего интерфейс *IClassFactory*

*CFunction.h (продолжение)*

```
class CFunctionClassFactory : public IClassFactory {
public:
    CFunctionClassFactory(){
        m_nRef = 0;
        g_cLock++;
        printf("Class factory object created\n");
    }

    ~CFunctionClassFactory(){
        g_cLock--;
        printf("Class factory object destroyed\n");
    }

    // методы наследуемые от IUnknown
    STDMETHOD(QueryInterface)(REFIID, void**);
    STDMETHOD_(ULONG, AddRef)();
    STDMETHOD_(ULONG, Release)();

    // методы наследуемые от IClassFactory
    STDMETHOD(CreateInstance)(LPUNKNOWN, REFIID, void**);
    STDMETHOD(LockServer)(BOOL);
protected:
    ULONG m_nRef; // счетчик ссылок
};
```

*guid.h*

```
// {68D8109B-A9E9-4D22-ACA8-F10A6D31B3aa}
static const CLSID CLSID_IFunction =
{ 0x68d8109b, 0xa9e9, 0x4d22, { 0xac, 0xa8, 0xf1, 0xa, 0x6d, 0x31, 0xb3, 0xaa } };

// {9A3EA520-1FF2-45DD-9C0D-97AEA1D6B9aa}
static const IID IID_IFunction =
{ 0x9a3ea520, 0x1ff2, 0x45dd, { 0x9c, 0xd, 0x97, 0xae, 0xa1, 0xd6, 0xb9, 0xaa } };

// {10B4565D-2794-4932-A0E2-EBB6C2CEE1aa}
static const IID IID_ISerialization =
{ 0x10b4565d, 0x2794, 0x4932, { 0xa0, 0xe2, 0xeb, 0xb6, 0xc2, 0xce, 0xe1, 0xaa } };
```

*CFunction.cpp*

```
#include <windows.h>
#include <objbase.h>
#include <stdio.h>
#include "guid.h"
#include "CFunction.h"
```

```
ULONG g_cObj=0;
ULONG g_cLock=0;
```

## Реализация виртуальных методов, унаследованных от *IUnknown*

*CFunction.cpp (продолжение)*

```
STDMETHODIMP
CFunction::QueryInterface(REFIID iid, void** ppv){
    if (iid == IID_IUnknown)
        *ppv = (IFunction*) this;
    else if (iid == IID_IFunction)
        *ppv = (IFunction*) this;
    else if (iid == IID_ISerialization)
        *ppv = (ISerialization*) this;
    else {
        *ppv = NULL;
        return E_NOINTERFACE;
    }
    AddRef();
    return NOERROR;
}

STDMETHODIMP_(ULONG) CFunction::AddRef(){
    return ++m_nRef;
}

STDMETHODIMP_(ULONG) CFunction::Release(){
    if(--m_nRef == 0){
        --g_cObj;
        delete this;
        printf("Object destroyed\n");
        return 0;
    }
    return m_nRef;
}
```



## Реализация виртуальных методов, унаследованных от IFunction

*CFunction.cpp (продолжение)*

```
STDMETHODIMP CFunction::getValue(double* pd){
    *pd = d;
    printf("It throws out %g\n", d);
    return S_OK;
}

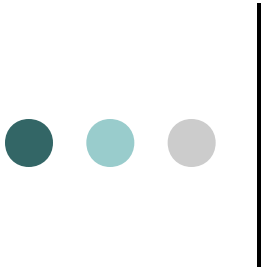
STDMETHODIMP CFunction::setValue(double D){
    d = D;
    printf("It swallows %g\n", d);
    return S_OK;
}
```

## Реализация виртуальных методов, унаследованных от ISerialization

*CFunction.cpp (продолжение)*

```
STDMETHODIMP CFunction::Load(){
    printf("It loads data\n"); return S_OK;
}

STDMETHODIMP CFunction::Store(){
    printf("It stores data\n"); return S_OK;
}
```



## Реализация в классе *CFunctionClassFactory* виртуальных методов, унаследованных от *IUnknown*

```
STDMETHODIMP
CFunctionClassFactory::QueryInterface(REFIID riid, void** ppv) {
    *ppv = NULL;

    if (IID_IUnknown == riid || IID_IClassFactory == riid)
        *ppv = this;
    if (NULL != *ppv){
        ((LPUNKNOWN)*ppv)->AddRef();
        return NOERROR;
    }
    return E_NOINTERFACE;
}
```

```
STDMETHODIMP_(ULONG) CFunctionClassFactory::AddRef() {
    return ++m_nRef;
}
```

```
STDMETHODIMP_(ULONG) CFunctionClassFactory::Release() {
    if(--m_nRef == 0){
        delete this;
        return 0;
    }
    return m_nRef;
}
```





## Реализация виртуальных методов, унаследованных от IClassFactory

```
STDMETHODIMP CFunctionClassFactory::CreateInstance(LPUNKNOWN pUnkOuter, REFIID riid, void**  
                                                    ppvObj) {  
    CFunction* pObj;  
    HRESULT hr;  
    *ppvObj = NULL;  
    hr = E_OUTOFMEMORY;  
  
    pObj = new CFunction;  
    if (NULL == pObj) return hr;  
  
    hr = pObj->QueryInterface(riid, ppvObj);  
    if (FAILED(hr))  
        delete pObj;  
    else  
        g_cObj++;  
    return hr;  
}  
STDMETHODIMP CFunctionClassFactory::LockServer(BOOL fLock) {  
    if (fLock) g_cLock++;  
    else      g_cLock--;  
  
    return NOERROR;  
}
```



## Точка входа в *dll*, реализующую компоненту

*just\_for\_fun.cpp*

```
#include <windows.h>
#include <objbase.h>
#include <stdio.h>
#include "guid.h"
#include "CFunction.h"

extern ULONG g_cObj;
extern ULONG g_cLock;

extern "C"
BOOL WINAPI DllMain(HINSTANCE hInstance, DWORD dwReason, LPVOID /*lpReserved*/){
    if (dwReason == DLL_PROCESS_ATTACH){
        printf("Library loaded\n");
    }
    else if (dwReason == DLL_PROCESS_DETACH){
        printf("Library unloaded\n");
    }
    return TRUE;    // ok
}
```



## Экспортируемые функции для вызова системными функциями COM

```
STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, void** ppv) {  
    HRESULT hr;  
    CFunctionClassFactory *pObj;  
  
    if (CLSID_Function != rclsid)    return E_FAIL;  
  
    pObj = new CFunctionClassFactory();  
  
    if (NULL==pObj)                return E_OUTOFMEMORY;  
  
    hr=pObj->QueryInterface(riid, ppv);  
  
    if (FAILED(hr))    delete pObj;  
  
    return hr;  
}
```

```
STDAPI DllCanUnloadNow() {  
    SCODE sc;  
    if (g_cObj == 0 && g_cLock == 0)  
        sc = S_OK;  
    else  
        sc = S_FALSE;  
  
    return sc;  
}
```

```
#include <windows.h>
#include <stdio.h>
#include <objbase.h>
#include "guid.h"
#include "root.h"
int main(){
    IFunction* pif=NULL;
    ISerialization* pii=NULL;
    DWORD dwContext =
        CLSCTX_INPROC_SERVER | CLSCTX_LOCAL_SERVER;

    CoInitialize(NULL);
    HRESULT hr = CoCreateInstance( CLSID_IFunction, NULL, dwContext, IID_IFunction, (void**) &pif);
    if(!hr){
        pif->setValue(3.141592);
        double d;
        pif->getValue(&d);
        printf("It's thrown out %g\n",d);

        hr=pif->QueryInterface(IID_ISerialization, (void**)&pii);
        pii->Load();
        pii->Store();
    }
    else
        printf("CoCreateInstance %i %x\n", hr, hr);
    pii->Release(); pif->Release(); CoUninitialize(); return 0;
}
```

## Клиент COM-сервера с использованием функции **CoGetClassObject**

*test2.cpp*

```
#include <windows.h>
#include <stdio.h>
#include <objbase.h>
#include "guid.h"
#include "root.h"

int main(){
    IClassFactory* pcf=NULL;
    IFunction* pii=NULL;
    ISerialization* pif=NULL;
    DWORD dwContext = CLSCTX_INPROC_SERVER | CLSCTX_LOCAL_SERVER;

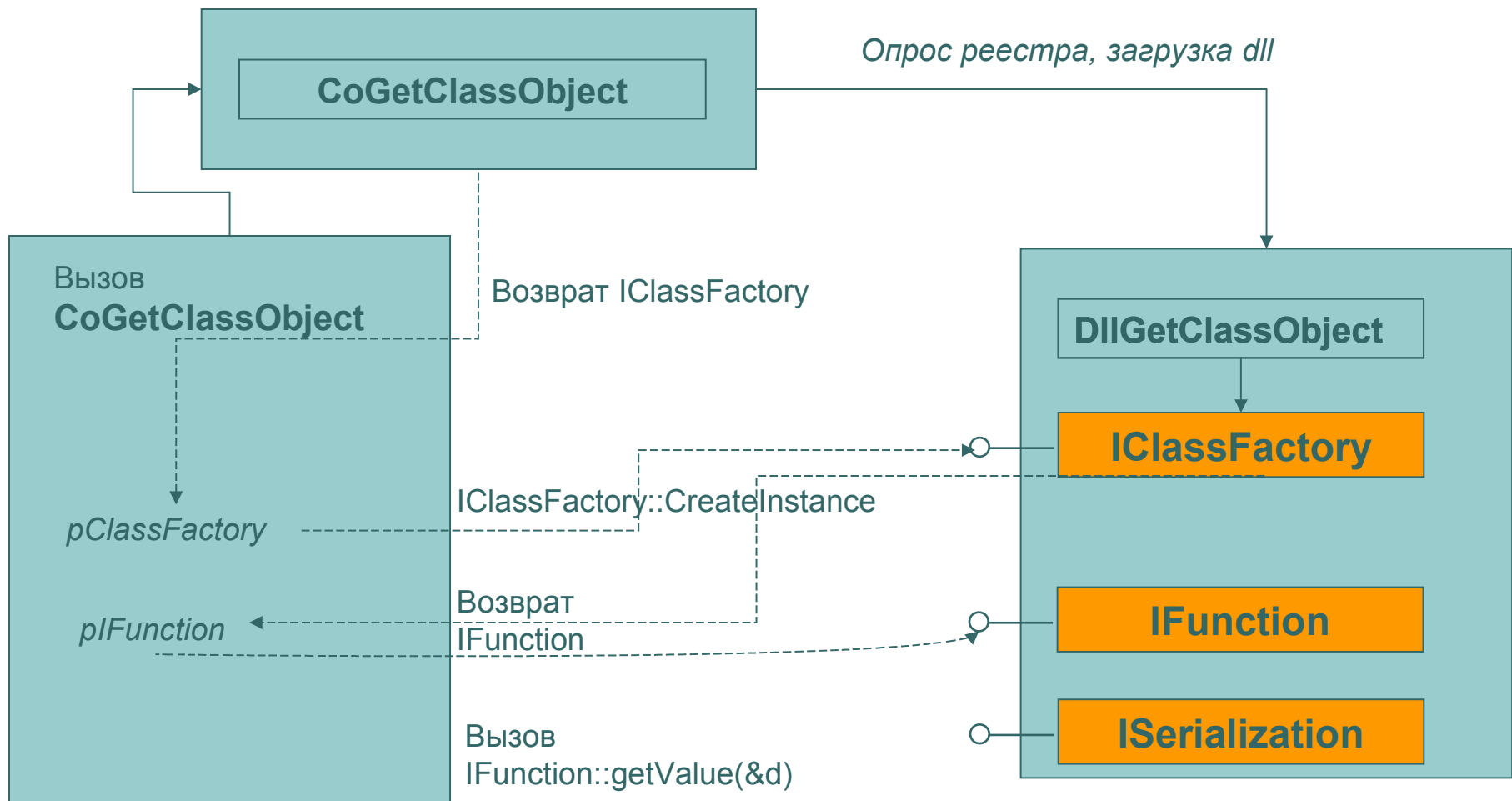
    CoInitialize(NULL);
    HRESULT hr = CoGetClassObject( CLSID_IFunction, dwContext, NULL, IID_IClassFactory, (void**) &pcf);
    pcf->LockServer(TRUE);
    pcf->CreateInstance(NULL, IID_IFunction, (void**)&pii);

    pii->setValue(3.141592);
    double d;
    pii->getValue(&d);
    printf("It's thrown out %g\n", d);

    pii->QueryInterface(IID_ISerialization, (void**)&pif);
    pif->Load();
    pif->Store();

    pii->Release(); pif->Release(); pcf->LockServer(FALSE); pcf->Release(); CoUninitialize(); return 0;
}
```

## Последовательность создания компонента клиентом





## Компиляция *dll*, реализующей *COM*-сервер

```
> cl /c just_for_fun.cpp CFunction.cpp
```

```
> link /DLL /DEF:just_for_fun.def just_for_fun.obj CFunction.obj ole32.lib
```

### *just\_for\_fun.def*

```
; just_for_fun.def  
  
LIBRARY      just_for_fun  
DESCRIPTION  'just_for_fun DLL Server'  
  
EXPORTS  
DllCanUnloadNow    @1 PRIVATE  
DllGetClassObject  @2 PRIVATE
```

## Компиляция клиента *COM*-сервера

```
> cl test.cpp ole32.lib
```



## Регистрация COM-сервера

*just\_for\_fun.reg*

```
REGEDIT
HKEY_CLASSES_ROOT\CLInput.MyInput\CLSID = {68D8109B-A9E9-4D22-ACA8-F10A6D31B360}
HKEY_CLASSES_ROOT\Wow6432Node\CLSID\{68D8109B-A9E9-4D22-ACA8-F10A6D31B360} =
CFunction MyFunction DLL
HKEY_CLASSES_ROOT\Wow6432Node\CLSID\{68D8109B-A9E9-4D22-ACA8-
F10A6D31B360}\InprocServer32 = C:\run\just_for_fun.dll
HKEY_CLASSES_ROOT\Wow6432Node\CLSID\{68D8109B-A9E9-4D22-ACA8-F10A6D31B360}\ProgId =
CFunction.MyFunction
HKEY_CLASSES_ROOT\Wow6432Node\Interface\{9A3EA520-1FF2-45DD-9C0D-97AEA1D6B97C} =
IFunction
HKEY_CLASSES_ROOT\Wow6432Node\Interface\{9A3EA520-1FF2-45DD-9C0D-
97AEA1D6B97C}\NumMethods = 2
HKEY_CLASSES_ROOT\Wow6432Node\Interface\{10B4565D-2794-4932-A0E2-EBB6C2CEE18D} =
ISerialization
HKEY_CLASSES_ROOT\Wow6432Node\Interface\{10B4565D-2794-4932-A0E2-
EBB6C2CEE18D}\NumMethods = 2
```



# Генерация GUID (*Globally Unique Identifier*), формат для заголовочного файла

> guidgen.exe

The screenshot displays the Microsoft Visual Studio Express 2013 IDE. The main window shows a C++ header file named `just_for_fun.h` with three GUID definitions for CLSID and IID. A semi-transparent text box with the text "Globally Unique Identifier" is overlaid on the code. On the right, the "Create GUID" dialog box is open, showing the "GUID Format" section with six options. Option 3, "static const struct GUID = { ... }", is selected. The "Result" section shows the generated GUID: `{10B4565D-2794-4932-A0E2-EBB6C2CEE18D}`.

```
// {68D8109B-A9E9-4D22-ACA8-F10A6D31B360}  
static const CLSID CLSID_IFunction =  
{ 0x68d8109b, 0xa9e9, 0x4d22, { 0xаc, 0xa8, 0xf1, 0xa, 0x6d, 0x31, 0xb3, 0x60 } };
```

```
// {9A3EA520-1FF2-45DD-9C0D-97AEA1D6B97C}  
static const IID IID_IFunction =  
{ 0x9a3ea520, 0x1ff2, 0x45dd, { 0x9c, 0xd, 0x97, 0xae, 0xa1, 0xd6, 0xb9, 0x7c } };
```

```
// {10B4565D-2794-4932-A0E2-EBB6C2CEE18D}  
static const IID IID_ISerialization =  
{ 0x10b4565d, 0x2794, 0x4932, { 0xa0, 0xe2, 0xeb, 0xb6, 0xc2, 0xce, 0xe1, 0x8d } };
```

**Create GUID**

Choose the desired format below, then select "Copy" to copy the results to the clipboard (the results can then be pasted into your source code). Choose "Exit" when done.

GUID Format

- ☐ 1. IMPLEMENT\_OLECREATE(...)
- ☐ 2. DEFINE\_GUID(...)
- ☒ 3. static const struct GUID = { ... }
- ☐ 4. Registry Format (ie. {xxxxxxxx-xxxx-xxxx-xxxx-xxxx})
- ☐ 5. [Guid("xxxxxxxx-xxxx-xxxx-xxxx-xxxx")]
- ☐ 6. <Guid("xxxxxxxx-xxxx-xxxx-xxxx-xxxx")>

Result

```
// {10B4565D-2794-4932-A0E2-EBB6C2CEE18D}  
static const GUID <<name>> =  
{ 0x10b4565d, 0x2794, 0x4932, { 0xa0, 0xe2, 0xeb, 0xb6, 0xc2, 0xce, 0xe1, 0x8d } };
```

# Генерация GUID (*Globally Unique Identifier*), формат для файла реестра

The image shows a screenshot of Microsoft Visual Studio Express 2013 for Windows Desktop. The main window displays the 'just\_for\_fun.reg' file, which contains registry entries for a COM class. The entries are as follows:

```
REGEDIT
HKEY_CLASSES_ROOT\CLSID\{68D8109B-A9E9-4D22-ACA8-F10A6D31B360} = CFunction MyFunction DLL
HKEY_CLASSES_ROOT\Wow6432Node\CLSID\{68D8109B-A9E9-4D22-ACA8-F10A6D31B360}\InprocServer32 = C:\run\just_for_f
HKEY_CLASSES_ROOT\Wow6432Node\CLSID\{68D8109B-A9E9-4D22-ACA8-F10A6D31B360}\ProgId = CFunction.MyFunction
HKEY_CLASSES_ROOT\Wow6432Node\Interface\{9A3EA520-1FF2-45DD-9C0D-97AEA1D6B97C} = IFunction
HKEY_CLASSES_ROOT\Wow6432Node\Interface\{9A3EA520-1FF2-45DD-9C0D-97AEA1D6B97C}\NumMethods = 2
HKEY_CLASSES_ROOT\Wow6432Node\Interface\{10B4565D-2794-4932-A0E2-EBB6C2CEE18D} = ISerialization
HKEY_CLASSES_ROOT\Wow6432Node\Interface\{10B4565D-2794-4932-A0E2-EBB6C2CEE18D}\NumMethods = 2
```

A 'Create GUID' dialog box is open on the right side of the screen. It prompts the user to choose a GUID format and then select 'Copy' to copy the results to the clipboard. The 'Registry Format' is selected, and the resulting GUID is displayed in the 'Result' field.

**Create GUID**

Choose the desired format below, then select "Copy" to copy the results to the clipboard (the results can then be pasted into your source code). Choose "Exit" when done.

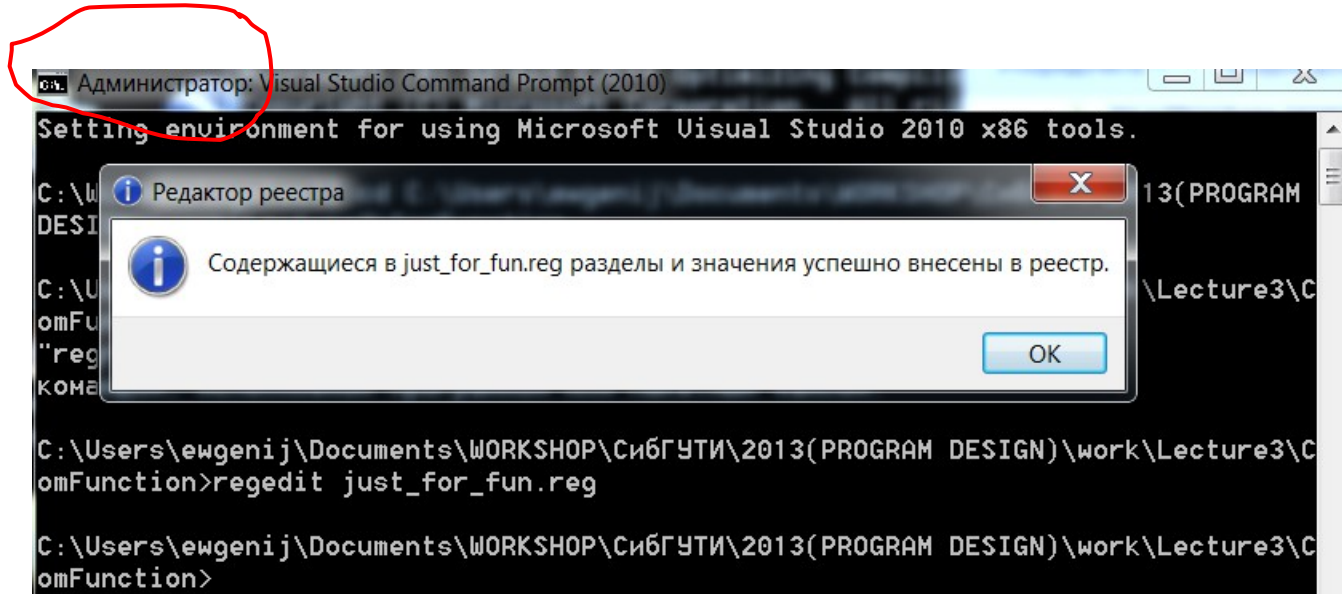
GUID Format

- ☐ 1. IMPLEMENT\_OLECREATE(...)
- ☐ 2. DEFINE\_GUID(...)
- ☐ 3. static const struct GUID = { ... }
- ☒ 4. Registry Format (ie. {xxxxxxxx-xxxx-xxxx-xxxx-xxxx})
- ☐ 5. [Guid("xxxxxxxx-xxxx-xxxx-xxxx-xxxx")]
- ☐ 6. <Guid("xxxxxxxx-xxxx-xxxx-xxxx-xxxx")>

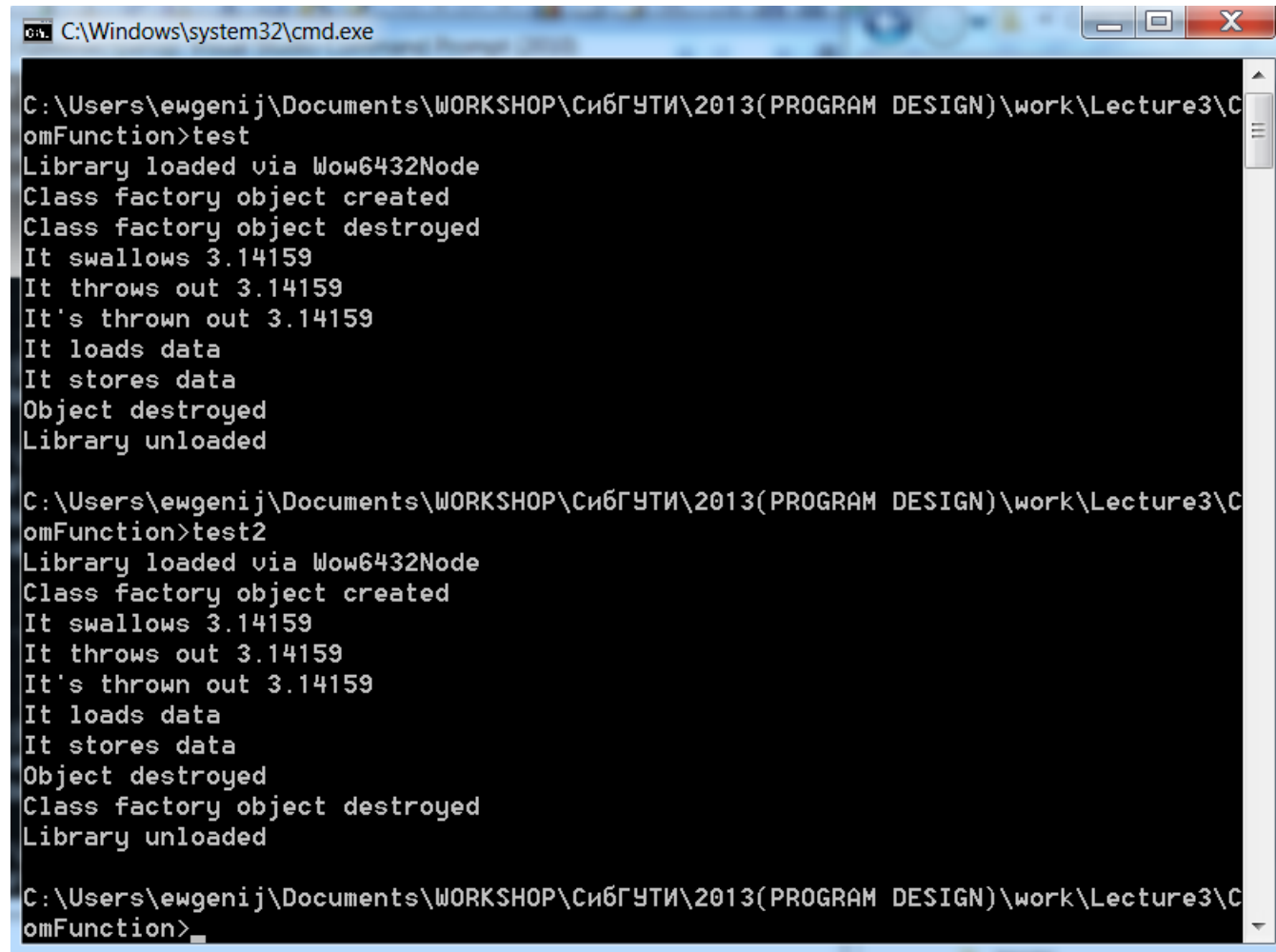
Result

{10B4565D-2794-4932-A0E2-EBB6C2CEE18D}

## Регистрация COM - компонента



## Тестирование COM - объекта



```
C:\Windows\system32\cmd.exe

C:\Users\ewgenij\Documents\WORKSHOP\СибГУТИ\2013(PROGRAM DESIGN)\work\Lecture3\ComFunction>test
Library loaded via Wow6432Node
Class factory object created
Class factory object destroyed
It swallows 3.14159
It throws out 3.14159
It's thrown out 3.14159
It loads data
It stores data
Object destroyed
Library unloaded

C:\Users\ewgenij\Documents\WORKSHOP\СибГУТИ\2013(PROGRAM DESIGN)\work\Lecture3\ComFunction>test2
Library loaded via Wow6432Node
Class factory object created
It swallows 3.14159
It throws out 3.14159
It's thrown out 3.14159
It loads data
It stores data
Object destroyed
Class factory object destroyed
Library unloaded

C:\Users\ewgenij\Documents\WORKSHOP\СибГУТИ\2013(PROGRAM DESIGN)\work\Lecture3\ComFunction>
```

## Автоматизация (анонс следующей лекции)

1.js

```
var obj;  
obj=new ActiveXObject("ATLTest.TestObj1.1");  
obj.TestMethod1();
```

