KPU Lab 2 Plug-inn

Formål:

At opnå erfaring med udvikling af en plug-in arkitektur i C++.

Forudsætninger

At du har læst artiklen fra MSDN om DLL-filer samt har kendskab til Factory Method design mønstret.

Hjælp

Hvis du ikke kan huske design mønstret Factory Method kan du evt. studere factory-eksemplerne fra Bruce Eckel Download HER, og du kan evt. læse denne artiklel fra MSDN FactoryDesignPattern.pdf.

Gode artikler om plug-ins:

- http://www.abstraction.net/ViewArticle.aspx?articleID=67
- http://www.nuclex.org/articles/cxx/4-building-a-better-plugin-architecture
- Plug-in Architecture Framework for Beginners (er skrevet til en noget ældre version af Visual Studio)

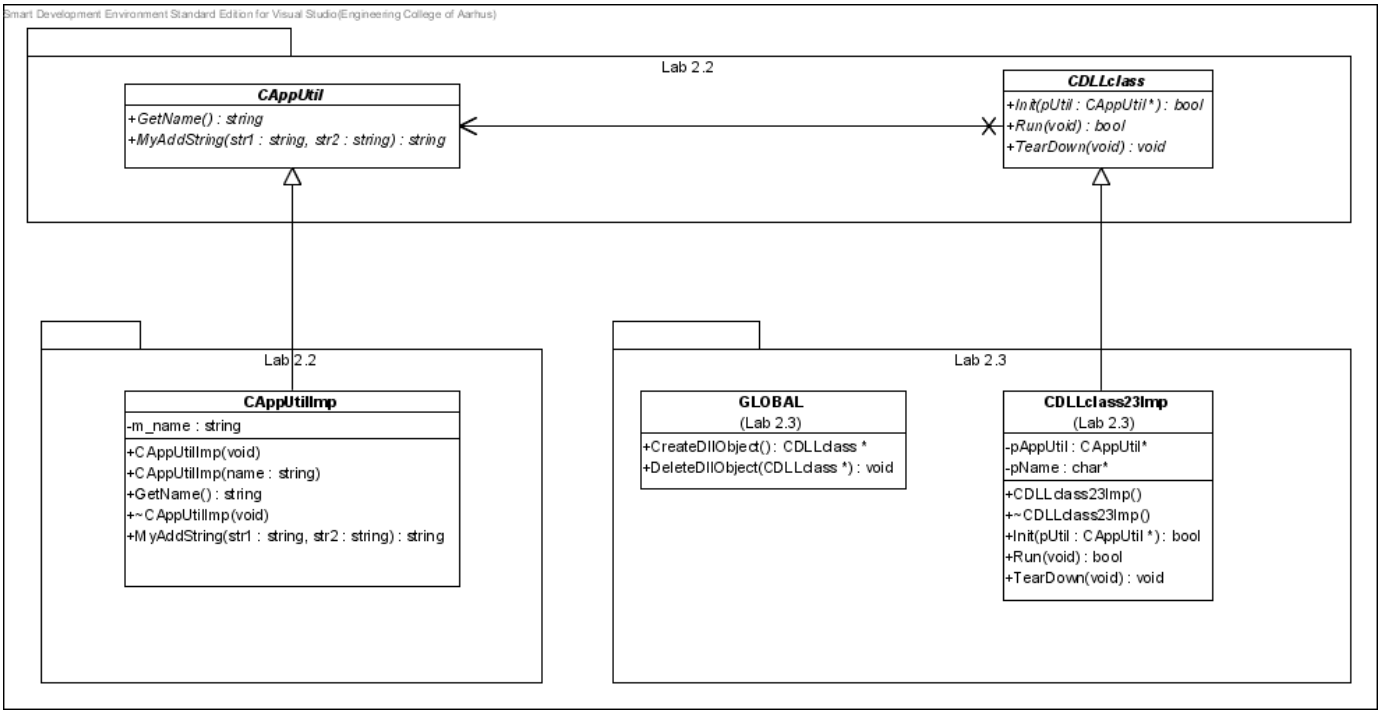
Delopgave 2.1:

Lav grænsefladen mellem applikationen og plug-in udviddelserne.

1. Fastlæg navn og signatur for den funktion som applikationen skal kalde for at få lavet en instans af hovedklassen i DLL'en (en Factory Method).
   Den kan f.eks. hedde CreateDllObject og have signaturen `CDLLclass * CreateDllObject ();`
   Ligeledes skal der også specificeres en funktion som skal bruges til at nedlægge objektet(-erne) igen.
   Den kan f.eks. hedde DeleteDllObject og have signaturen `void DeleteDllObject (CDLLclass *);`

2. Design den abstrakte basis klasse som hovedklassen i DLL'en skal nedarve fra.
   F.eks.:
   ```
   class CDLLclass
   {
   public:
     virtual bool Init(CAppUtil * pUtil) =0;
     virtual bool Run() =0;
     virtual void TearDown() =0;
   }
   ```

3. Design den abstrakte klasse som DLL'en modtager en pointer til i funktionen Init.
   F.eks.
   ```
   class CAppUtil
   {
   public:
     virtual string GetName() =0;
     virtual string MyAddString(string str1, string str2) =0;
   }
   ```



Delopgave 2.2:

Lav applikationen.

1. Implementer en klasse som nedarver fra CAppUtil (denne klasse bruges ikke af hovedprogrammet, men kan kaldes fra DLL'en via call-back).

2. Lav hovedprogrammet i applikationen således at det spørger brugeren om navnet (evt. inklusiv en sti) på den DLL-fil som applikationen skal loade og derefter kalde funktionerne Init, Run og TearDownpå DllObjektet i denne DLL.

Hint: **`LoadLibrary()`** kræver en parameter af typen LPCWSTR (Long Pointer Const Wide STRing) **->** vi skal bruge Unicode. I C++ kan man skifte til at arbejde med Unicode ved at sætte w foran: f.eks. wstring, og wcin. Streng-konstanter kan konverteres til Unicode med macroen TEXT("streng").

Delopgave 2.3:

Lav en plug-in DLL.

1. Lav en dll som eksporterer funktionerne: `CreateDllObject og DeleteDllObject.`
   `CreateDllObject` returnerer en instans af en klasse som nedarver fra CDLLclass. Du bestemmer selv hvad implementeringen af funktionerne skal lave, men mindst en af dem skal bruge en funktion fra CAppUtil.

2. Test at din applikation kan loade og kalde DLL'en.

Delopgave 2.4:

Lav en ny plug-in DLL.

1. Lav en ny dll som overholder det samme interface som i delopgave 2.3, men hvor implementeringerne af funktionerne laver noget andet.

2. Test at din applikation (uden at der er lavet ændringer i den) også kan loade og kalde den nye DLL'en.

Tilbage