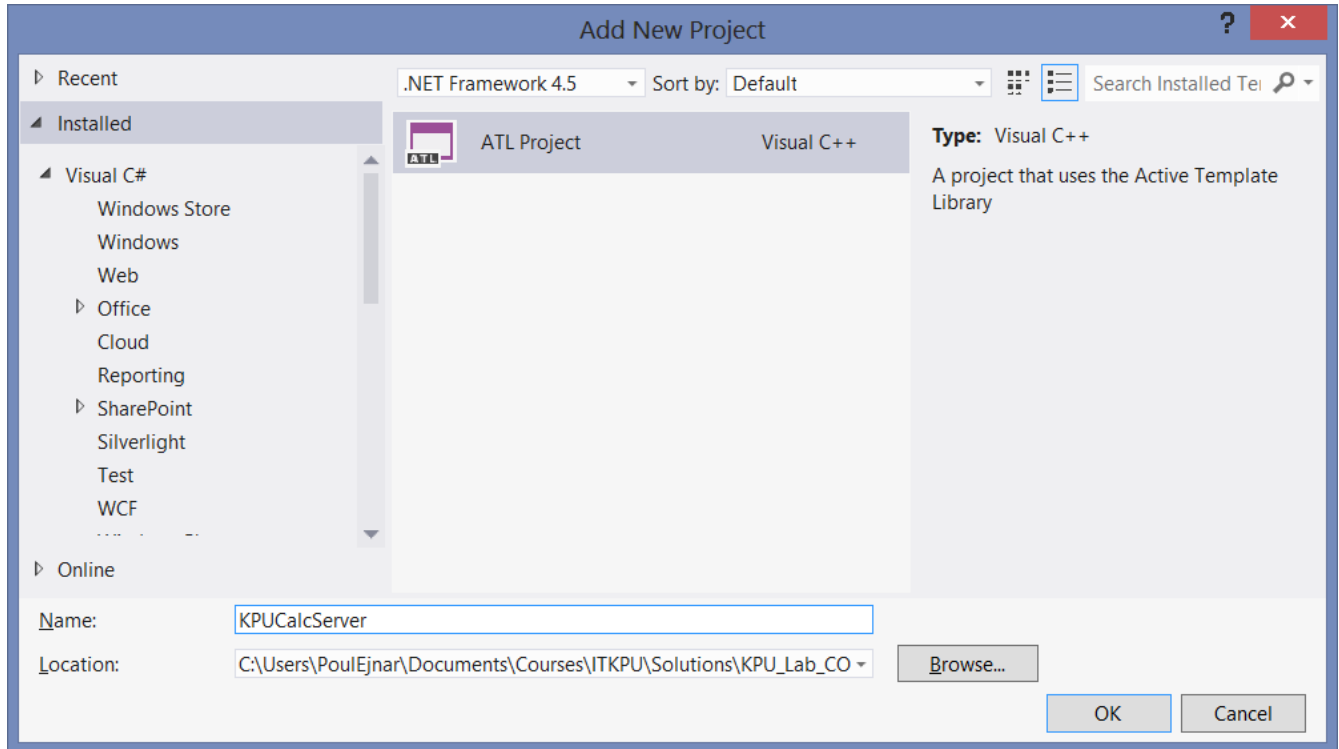


# ITKPU Løsningsforslag til Com med ATL

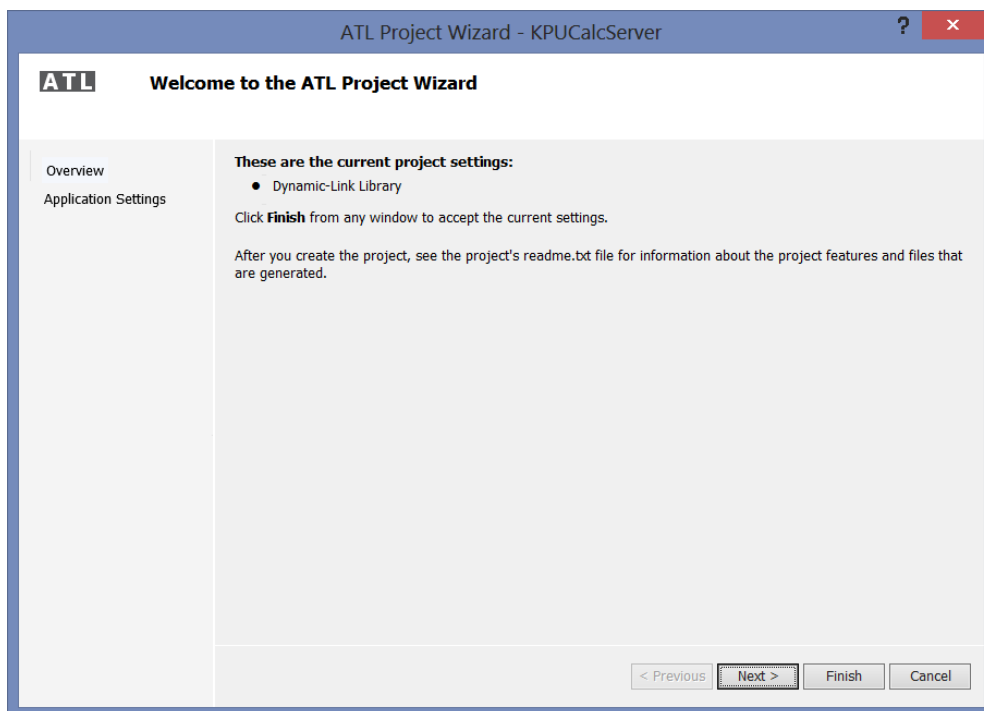
## Delopgave 1

Løsningsforslaget til delopgave 1 kan ses i projektet "KPUCalcServer".

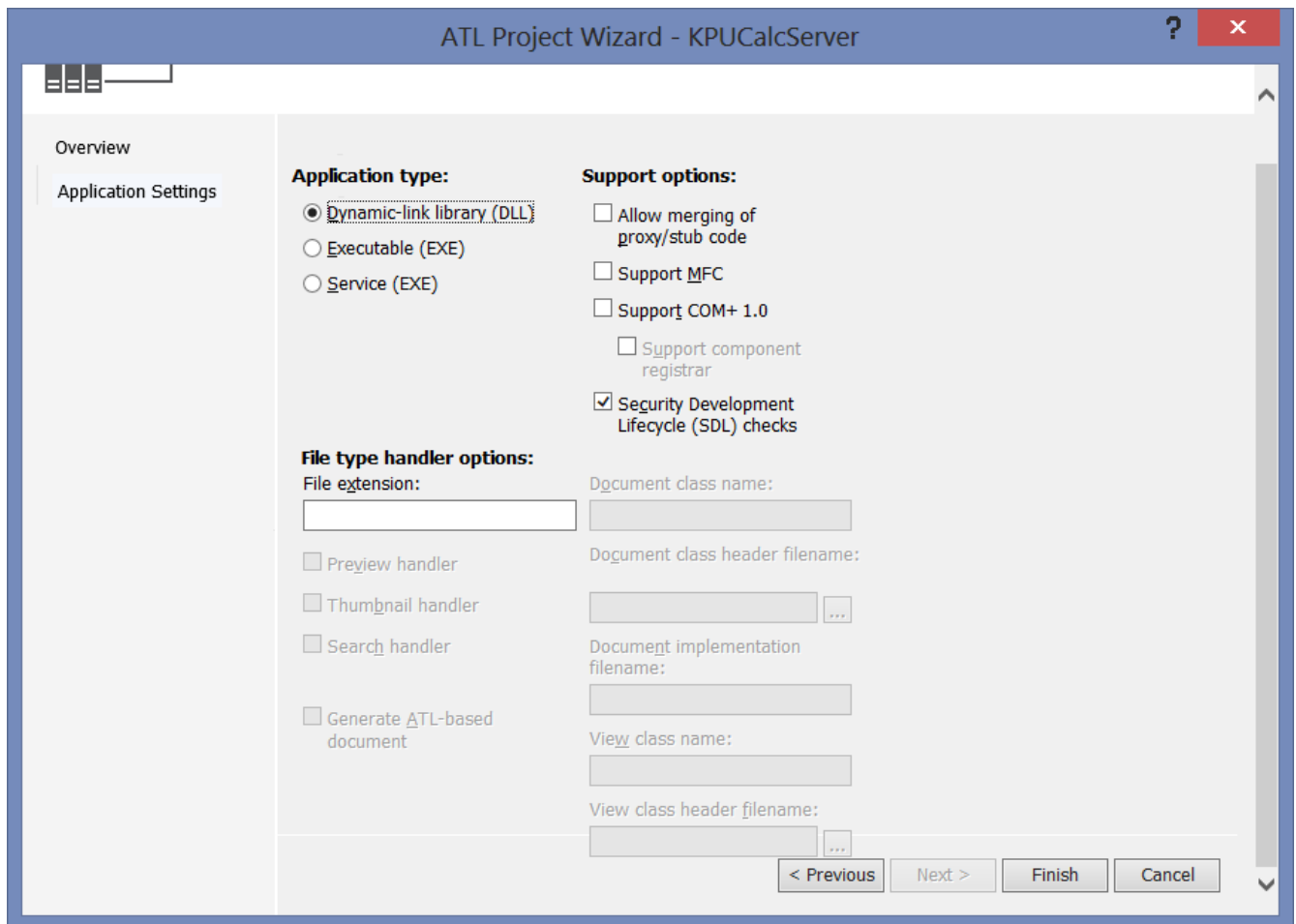
Jeg har valgt at lave et c++ - ATL Project (se figur) med følgende settings:



Klik på **Ok**:

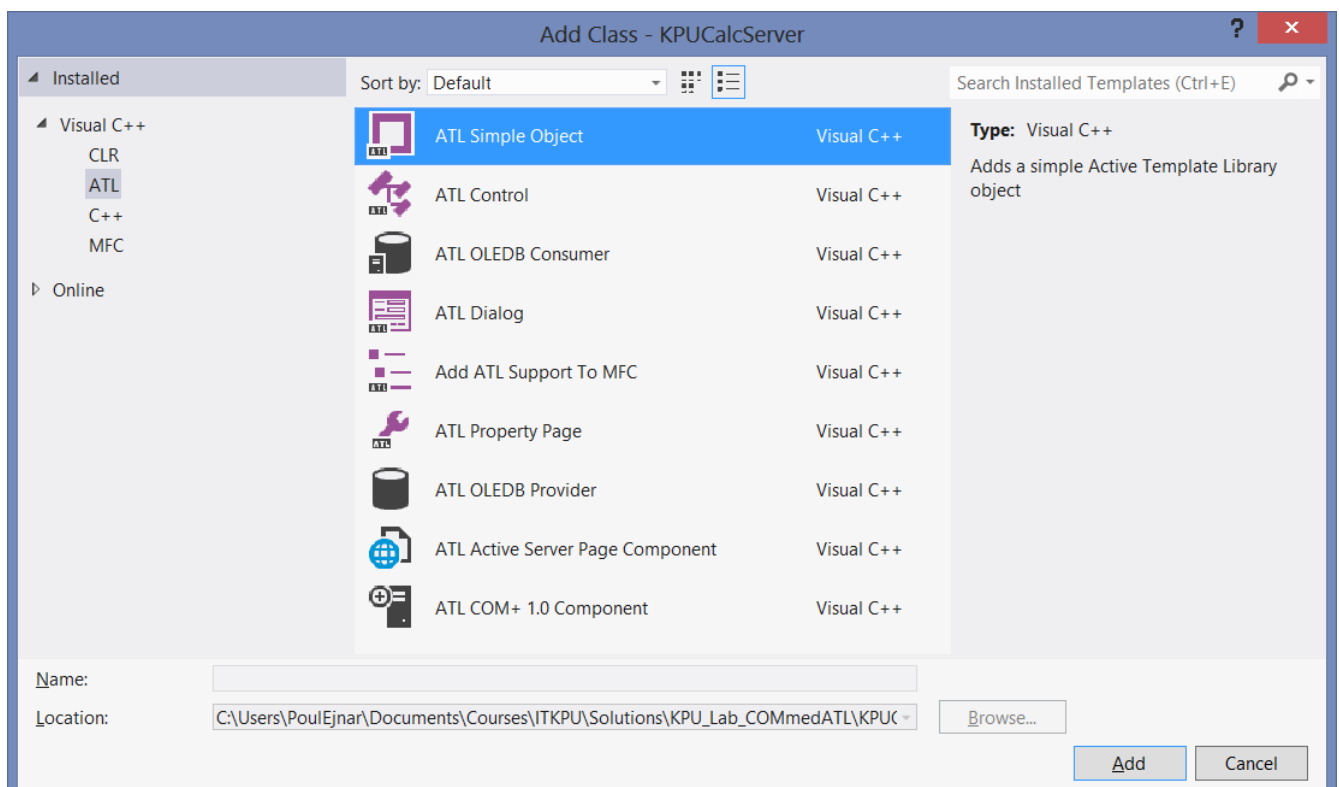


Klik på **Next**:



Klik på **finish**.

Herefter tilføjes (ved højreklik på projektet og valg af Add Class) et ATL Simple Object.



Klik på **Add**.

ATL Simple Object Wizard - KPU CalcServer

Welcome to the ATL Simple Object Wizard

Names  
File Type Options  
Options

**C++**

Short name: Calculator .h file: Calculator.h  
Class: CCalculator .cpp file: Calculator.cpp

**COM**

Coclass: Calculator Type: Calculator Class  
Interface: ICalculator ProgID: AU.ASE.KPU.Calculator

< Previous Next > Finish Cancel

Indtaster navn under "C++ **Short name**" og accepterer de øvrige genererede navne. Indtaster selv **ProgId**.

Klik på **Next**.

ATL Simple Object Wizard - KPU CalcServer

File Type Handler Options

Names  
File Type Options  
Options

☐ Add file type (preview/thumbnail/search handler) support

☒ Preview Handler File Extension:  
☐ Thumbnail Handler  
☐ Search Handler

**ATL-based Document**

Generic document class name: .h of generic document:

☐ Generate ATL-based document .cpp of generic document:

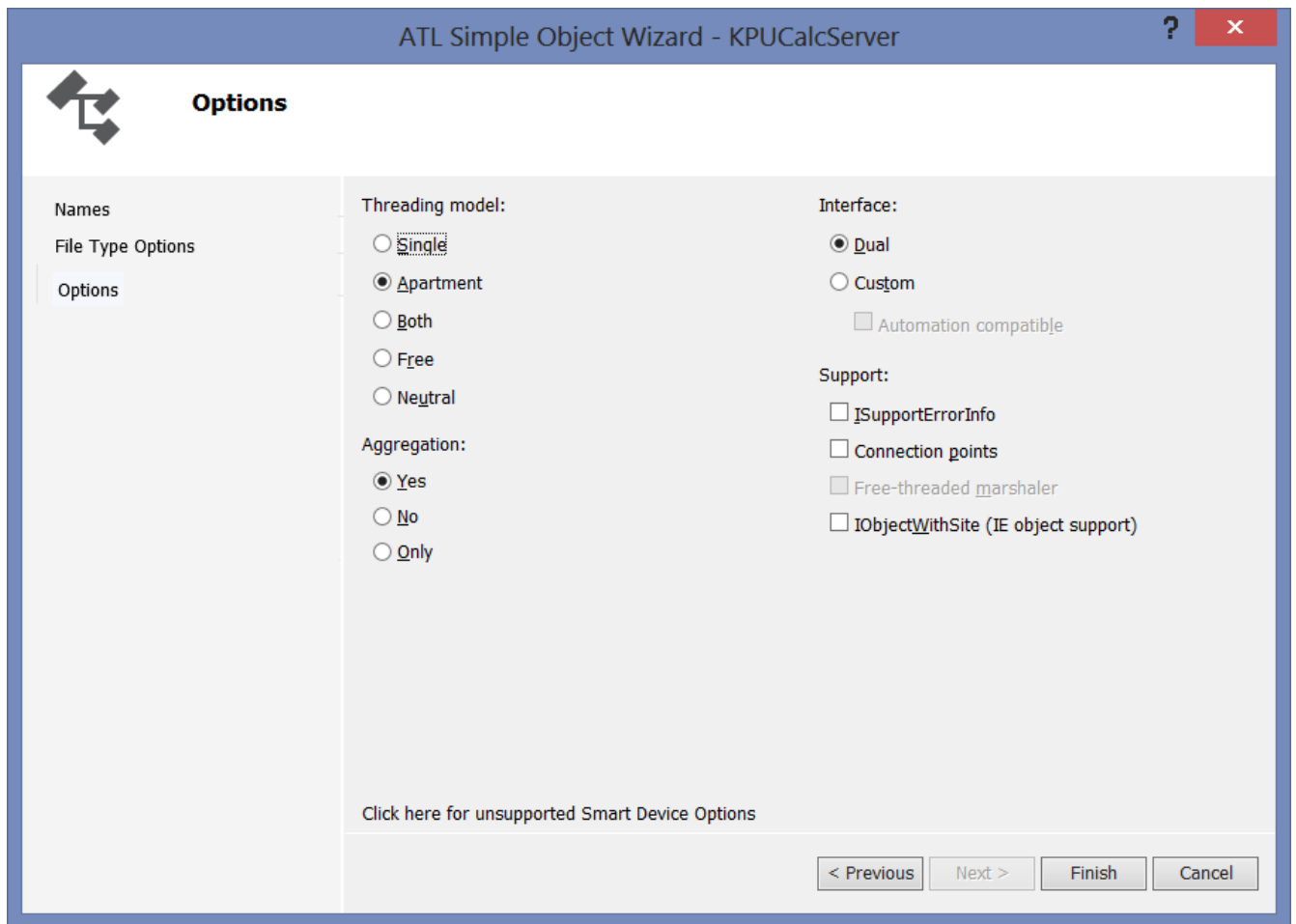
**MFC-based Document (and View)**

Document class name: .h of MFC document:

View class name: .h of view:

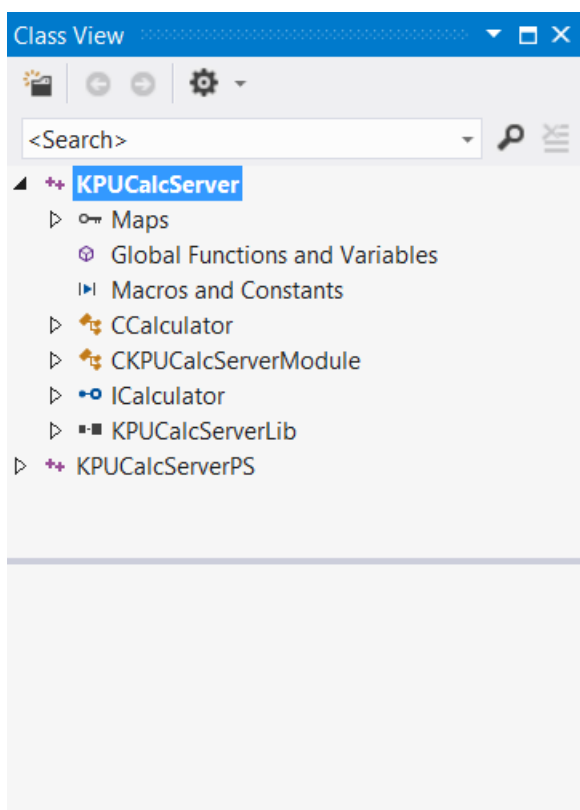
< Previous Next > Finish Cancel

Under File Type Options bruger jeg defaultindstillingerne.  
Klik på **Next**.



Bruger defaultindstillingerne. Ved tryk på **Finish** laver Visual Studio en ny klasse samt et interface.

Skifter til Class View og i Class View'et højre-klikker jeg på det interface, jeg lige har fået genereret - ICalculator, og vælger Add Method.



The screenshot shows the 'Add Method Wizard' dialog box for 'KPUCalcServer'. The title bar says 'Add Method Wizard - KPUCalcServer'. The main window has a logo and the text 'Welcome to the Add Method Wizard'. On the left, there are two tabs: 'Names' and 'IDL Attributes'. The 'Return type' is set to 'HRESULT'. The 'Method name' is 'Add'. Under 'Parameter attributes', there are checkboxes for 'in', 'out', and 'retval'. The 'Parameter type' is a dropdown menu. The 'Parameter name' is a text box. There are 'Add' and 'Remove' buttons. The parameter list shows three parameters: '[in] LONG arg1', '[in] LONG arg2', and '[out,retval] LONG\* result' (which is highlighted in blue). At the bottom, there are buttons for '< Previous', 'Next >', 'Finish', and 'Cancel'.

Angiver **method name** og tilføjer 3 **parametre** som vist på figuren, og klikker på **Next**:

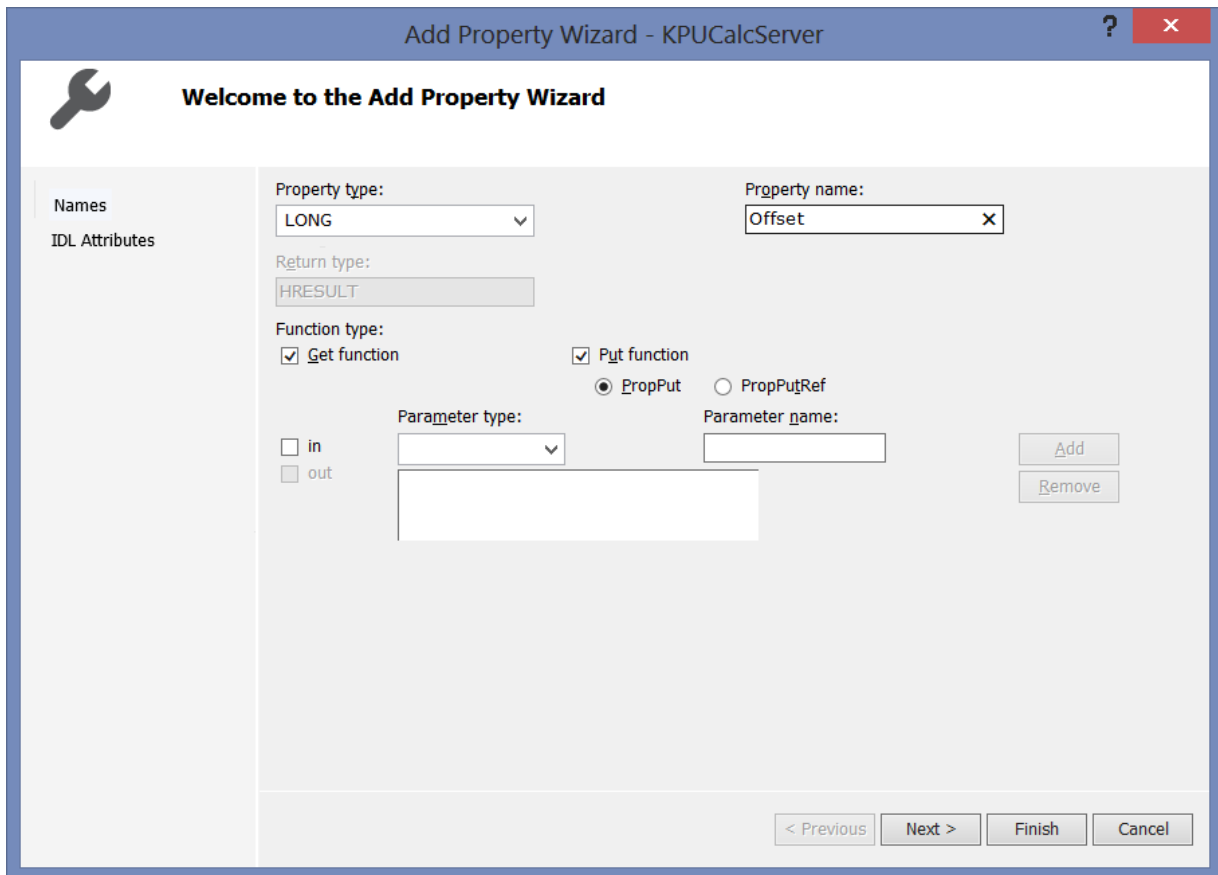
The screenshot shows the 'Add Method Wizard' dialog box for 'KPUCalcServer', now on the 'IDL Attributes' screen. The title bar says 'Add Method Wizard - KPUCalcServer'. The main window has a logo and the text 'IDL Attributes'. On the left, there are two tabs: 'Names' and 'IDL Attributes'. The 'id' is 'i'. The 'call\_as' is empty. There are checkboxes for 'hidden', 'source', 'local', 'restricted', and 'vararg'. The 'helpcontext' is empty. The 'helpstring' is empty. At the bottom, there are buttons for '< Previous', 'Next >', 'Finish', and 'Cancel'.

Og accepterer defaultværdierne med tryk på **Finish**.

I implementerings klassen Calculator.cpp implementerer jeg funktionen Add:

```
STDMETHODIMP CCalculator::Add(LONG arg1, LONG arg2, LONG* result)
{
    *result = arg1 + arg2 + offset;
    return S_OK;
}
```

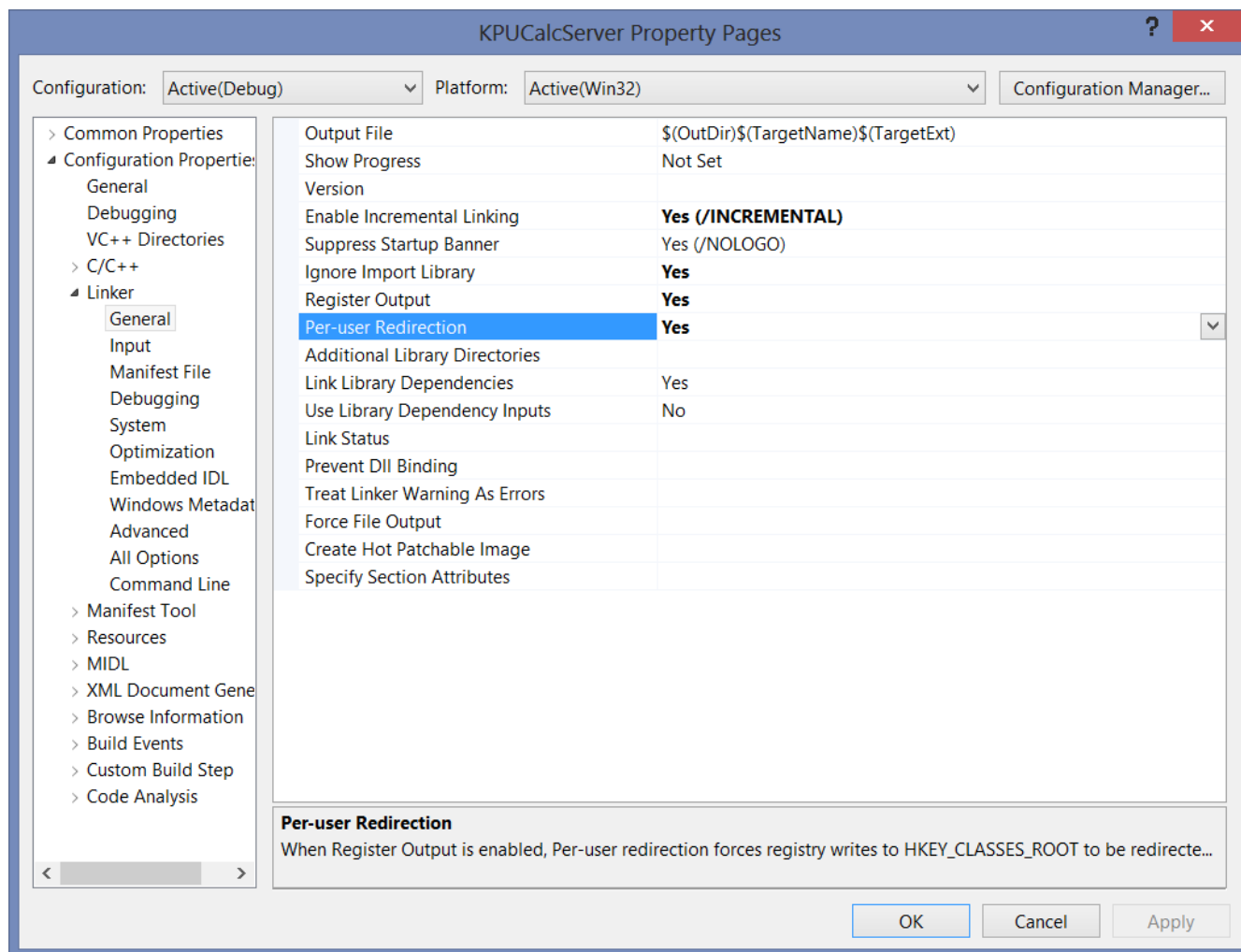
I header-filen tilføjer jeg et privat datamedlem af typen LONG og med navnet offset. Og husker at initialisere til 0 i konstruktor – bemærk, at denne er implementeret i headerfilen. På tilsvarende vis tilføjes og implementeres Sub-metoden og propertyen Offset.



Hermed er COM-serveren færdig og jeg vælger Build Solution i Build menuen. Herved oversættes source filerne og linkes til den færdig COM-server, som af VS automatisk bliver registreret. Bemærk, at ATL ikke bruger den simple metode med reg-filen, men indsætter kode i funktionerne DllRegisterServer() og DllUnRegisterServer().

### OBS VS's automatiske registrering kan fejle!

1. If you get a Project **Build Error PRJ0050** when you build a COM server then do either:
  - Run Visual Studio in elevated mode
  - Use Per-user Redirection Registration.  
(a Linker setting in Visual Studio).
    - Per-user redirection allows you to register without having to run in elevated mode.
2. *En anden mulig fejkilde er brug af specielle karaktere i path. Årsagen hertil er så vidt jeg kan se, at VS selv opbygger stien til COM-serveren, og dette giver en fejl hvis der er et "ø" i et af mappenavnene i stien til COM-serveren. Fejlen kan omgås ved at registrere COM-serveren manuelt med kommandoen: REGSVR32 KPUCalcServer.dll (gives i en Command Prompt).*

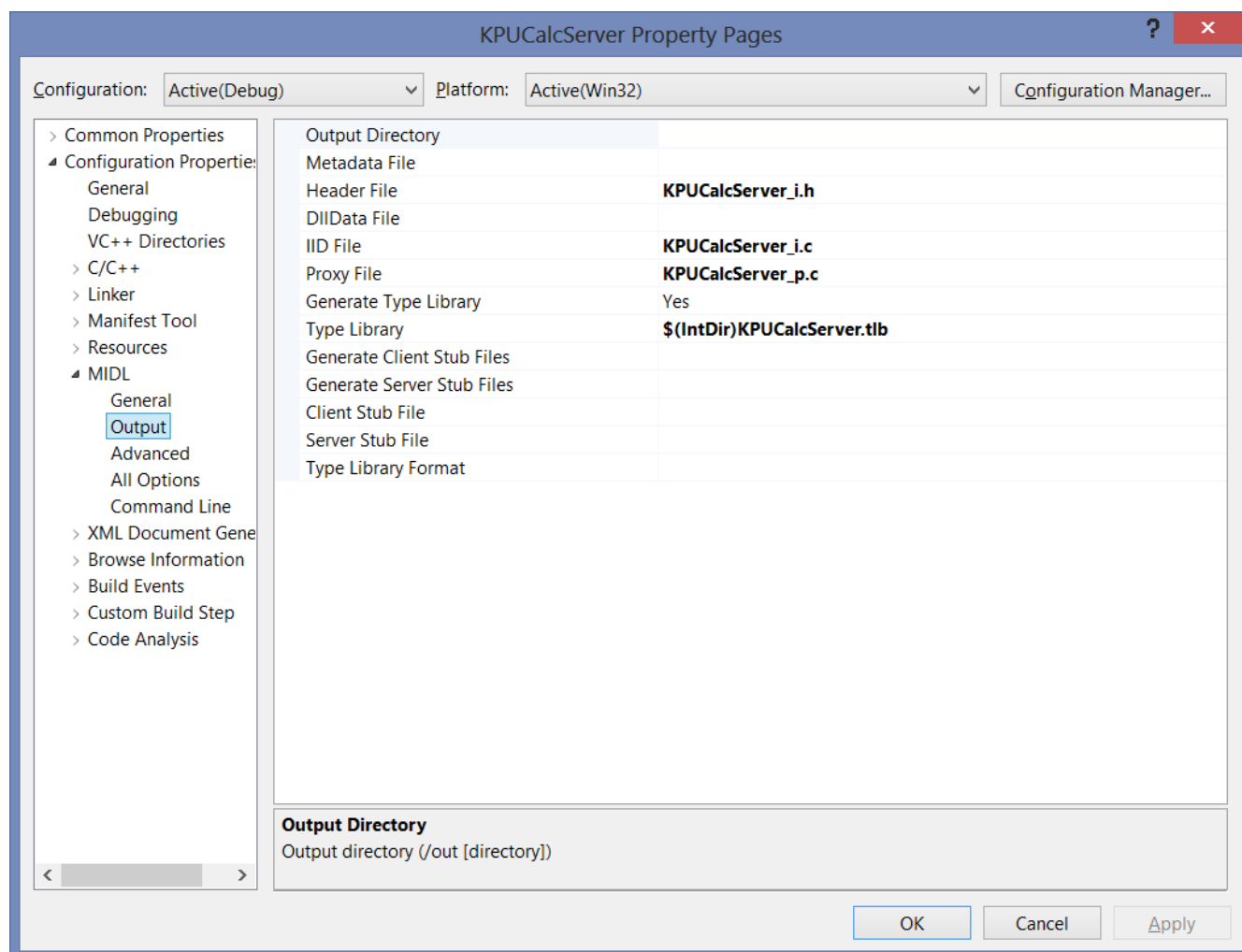


Bemærk, at jeg i filen Stdafx.h har indsat linie "#define \_ATL\_DEBUG\_INTERFACES" (skal indsættes før include-filerne). Denne makro får VS til at indsætte kode som udskriver debug info til output-vinduet, når COM-serveren køres i debug-tilstand, hvilket kan være en stor hjælp ved fejlfinding.

## Delopgave 2

Testprogrammet vælger jeg at lave som et console-baseret C++ program - dette kan ses i projektet SimpleCalcClient.

Wizard'en laver et nyt projekt og i dette findes en cpp-fil som jeg skriver min kode i. Jeg starter med at inkludere den nødvendige header-fil som erklærer COM-serverens interface. Bemærk, at det IKKE er implementerings klassens header fil, men den header-fil som MIDL genererer ud fra IDL-koden. Hvis man er i tvivl om hvilken fil det drejer sig om, så kan man højre-klikke på COM-server projektet og vælge properties. Klikke på MIDL - Output, hvorved man kan se navnet på den header-fil som MIDL genererer: KPUCalcServer\_i.h. Endvidere inkluderes den fil der indeholder Guid'ene (IID File): KPUCalcServer\_i.c. Disse inkluderes i cpp filen.



## Delopgave 3

Jeg har valgt at arbejde videre på projektet fra delopgave 1: KPUCalcServer.

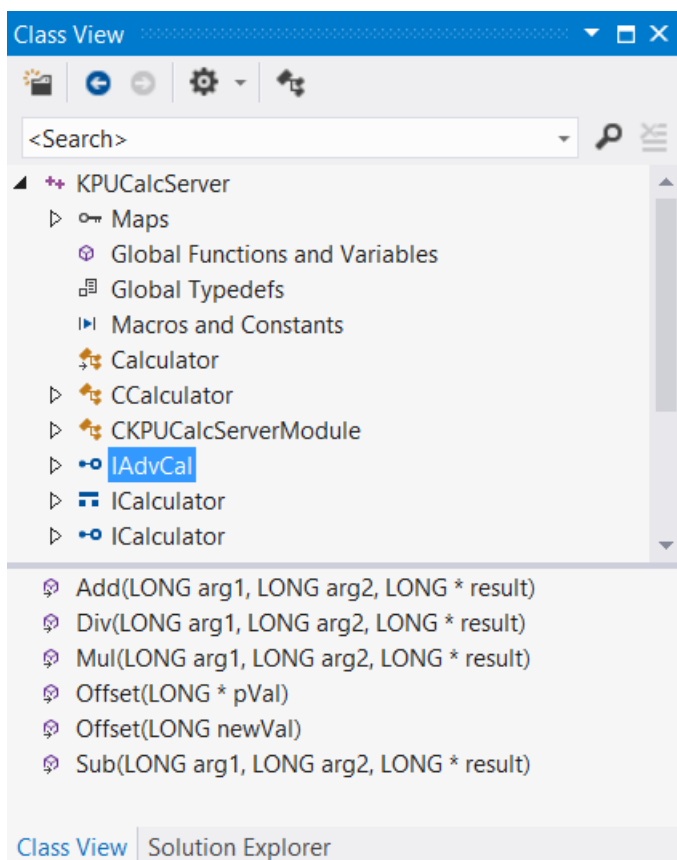
Jeg tilføjer et nyt interface ved at:

1. I IDL-filen kopiere ICalculator interface'et
2. I kopien ændrer jeg navnet til IAdvCal
3. Generere en ny guid som bruges til uuid for det nye interface
4. Tilføjer det nye interface til co-klassen:

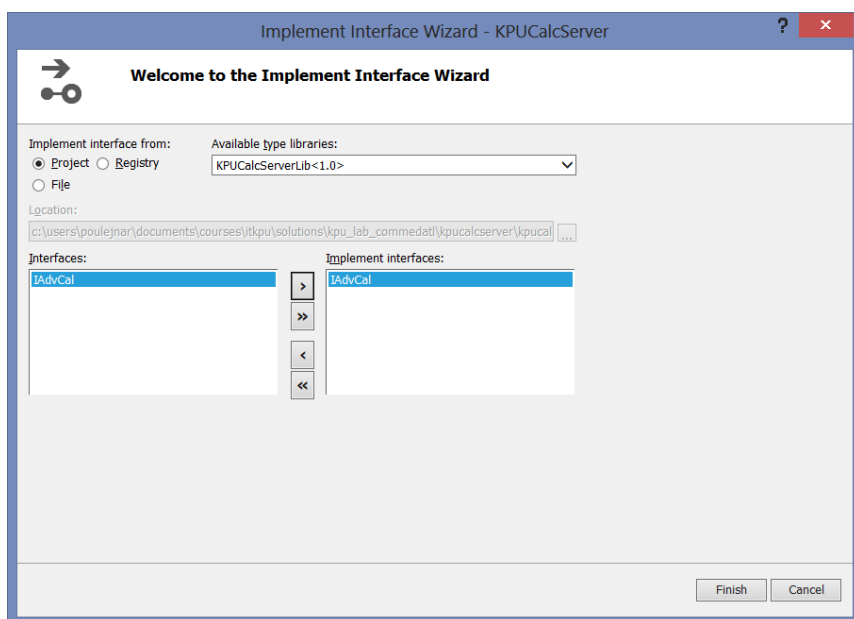
```
coclass Calculator
{
    [default] interface ICalculator;
    interface IAdvCal;
};
```



5. Efter at IDL-filen er blevet kompileret kan man så bruge wizard'en (ved at højreklikke på interface't i classview'et) til at tilføje de nye metoder.



Efter at det nye interface er defineret (og helst endeligt fastlagt) højreklikkes på klassen CCalculator, og der vælges add - implement interface. I den dialog der kommer op vælges det nye interface:

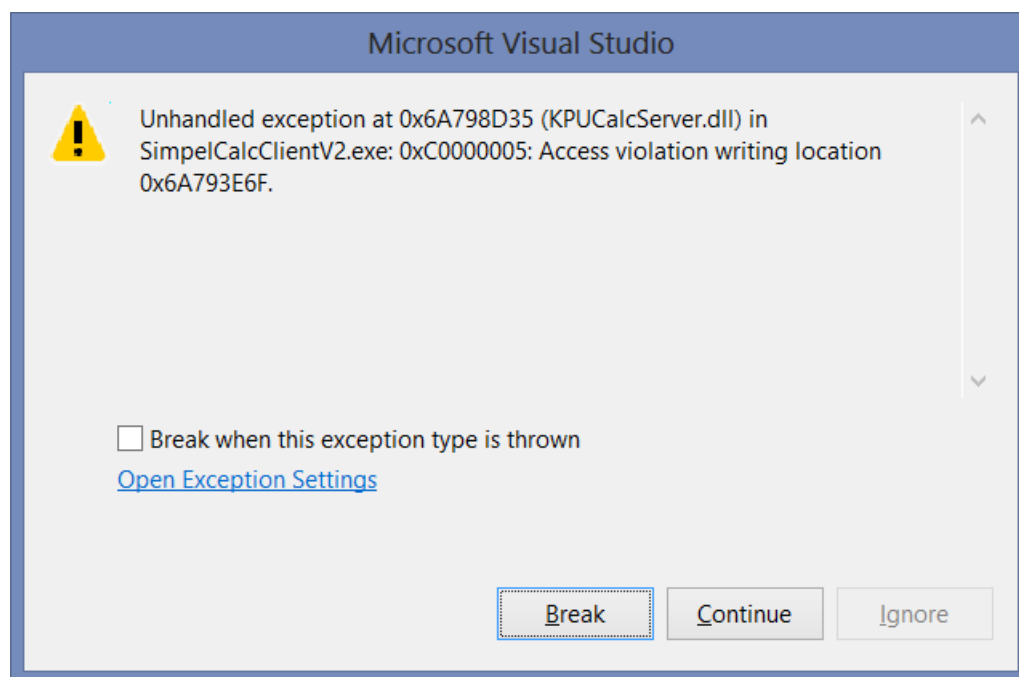


**Bemærk, at wizard'en kun indsætter kode for de nye funktioner i header-filen, man skal selv flytte implementeringen over i cpp-filen.**

## Delpgave 4

Jeg laver et nyt projekt (som i delopg. 2) som jeg kalder SimpleCalcClientV2. Kopierer koden fra delopg. 2, og laver de nødvendige tilretninger. Når man som her laver "copy and paste" programmering kommer man let til at lave nogle dumme fejl, fordi man ikke får ændret alt, hvad der skal ændres. F.eks. kan man undlade at ændre en IID\_id som vist i nedenstående eksempel:

```
HRESULT hr;
IAdvMath *pIAdvMath;
hr = CoCreateInstance( CLSID_Calculator, NULL, CLSCTX_INPROC_SERVER,
IID_ICalculator, (void**) &pIAdvMath);
// ^denne IID passer ikke til ^denne interface-pointer;
// men fejlen kommer først, når man langt senere kalder en funktion, som
// findes IAdvMath men ikke i ICalculator.
// Og fejlmeddelelsen er meget misvisende (det er fejlmeddelelser p.g.a.
// forkert typecast ofte)
```



## Ekstra

Jeg laver et nyt projekt (som i delopg. 2) som jeg kalder SmartCalcClient. I dette projekt bruger jeg smart pointere og ingen headerfil. I stedet for at bruge head'er filen fra lab 3 så importerer jeg type library, dog sker dette via .dll-filen, da ATL skriver type lib ind som en del af .dll filen med COM-serveren.

```
#import "..\\Debug\\KPUCalcServer.dll" no_namespace named_guids
```

Bemærk at intelliSense til at begynde med viser fejl i syntaksen, selv om der ikke er fejl. Man skal blot være "stærk i troen" og så bygge projektet. Efter et par builds er intelliSense med igen.

## Delopgave 5

Se løsningen hertil i filen Lab5.htm under Solution Items.

Ved kaldet til ActiveXObject er det ProgId som bruges. Og vær opmærksom på at det kun er metoderne i default interface interface, som kan kaldes fra JavaScript – og at funktionernes signatur er anderledes end når de kaldes fra C++.

## Delopgave x6

Se løsningen hertil i projektet KPUDynamicClient (dynamiske klienter i C++ er meget besværlige! - og derfor langt uden for pensum).