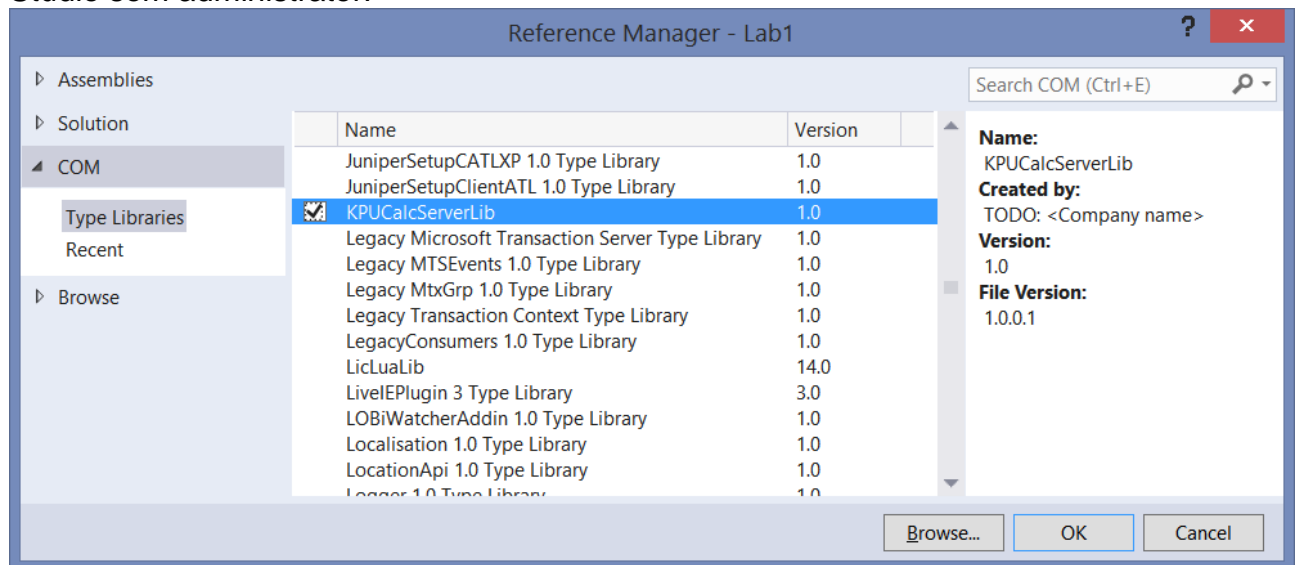


Vejledning til løsningsforslaget

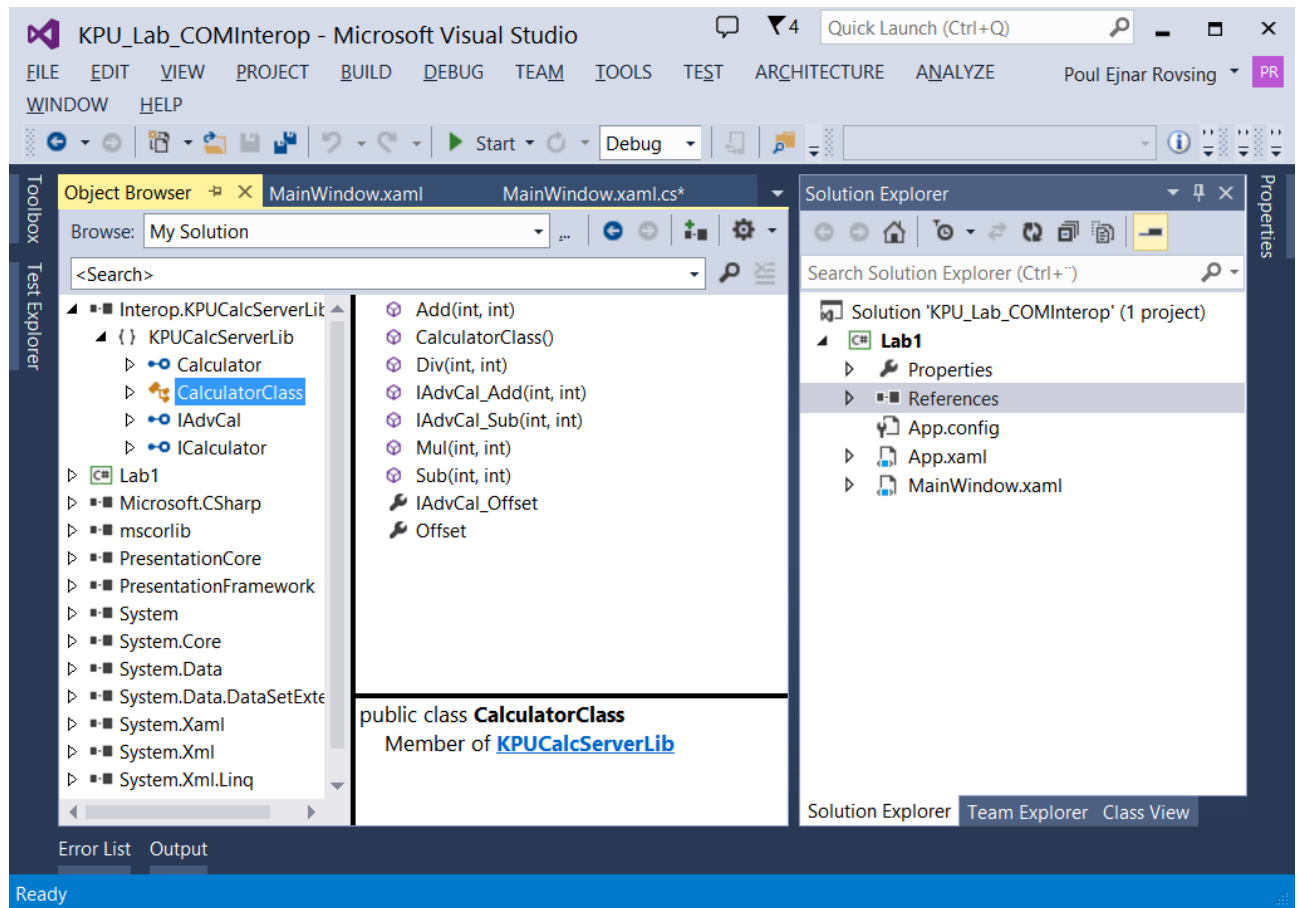
Delopgave 1

Løsningsforslaget hertil kan ses i projektet Calculator.

1. Før man laver en COM-klient i C# er det vigtigt at man er sikker på at COM-serveren virker, og at den er korrekt installeret!
2. Når man er sikker på det så laves et Windows WPF projekt som normalt.
3. Tilføj en reference til COM-serveren - dette virker kun når jeg IKKE kører Visual Studio som administrator!



4. I stedet for at gætte hvilke navne der er blevet genereret af TLBIMP, så åben Object Browseren (gøres i View-menuen).



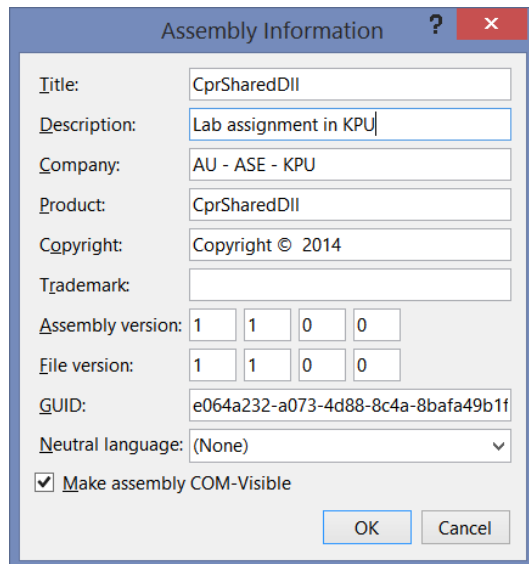
Her kan man så læse, at vi finder det vi har brugt for i namespace'et KPUCalcServerLib.

- Det anbefales at man koder op mod de samme interfaces som man ville gøre i en unmanaged COM client (ICalculator og IAdvCalc). Bemærk at det er det tomme interface Calculator som man skal kalde new på - noget underligt valg, da man normalt ikke kan kalde new på et interface. Alternativt kan man højreklikke på interop-assemblien i solution explorer'en og så sætte propertyen "Embed Interop Types" til false. Herefter kan man så kalde new på klassen CalculatorClass, og kode direkte op mod denne.

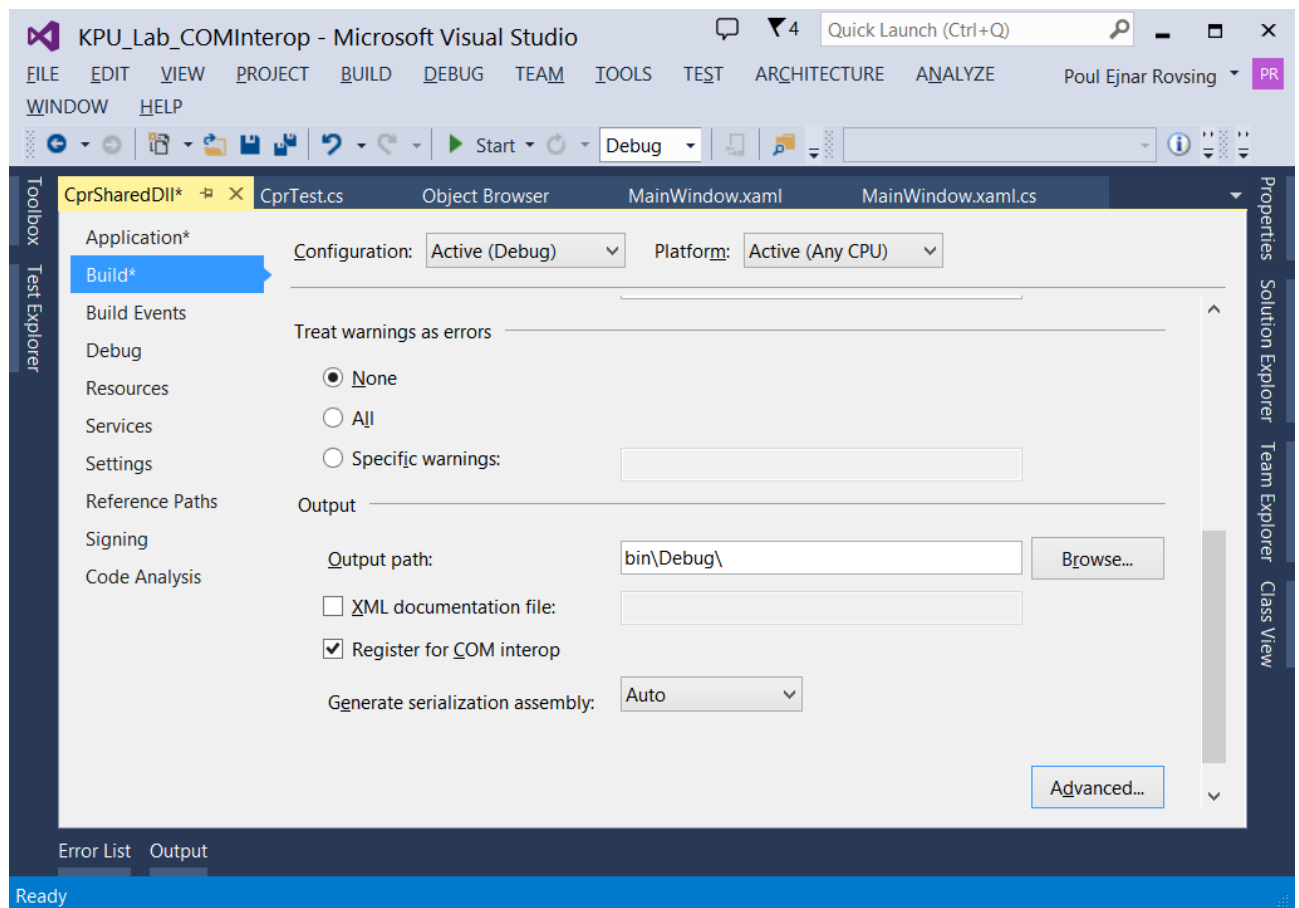
Delopgave 2

Her vises først den minimale måde at løse opgaven på. Denne løsning kan ses i projekterne CprSharedDll og CprComClient. **Man skal køre Visual Studio som administrator** (modsat lab1!) for at dll'en CprSharedDll kan registreres som en COM komponent.

- Kopierer projektet CprSharedDll til denne solutions mappe, og tilføj projektet til solution i Visual Studio (Add existing project).
- Under Project Settings – Assembly Information vælges "Make assembly COM visible" og assembly version ændres.

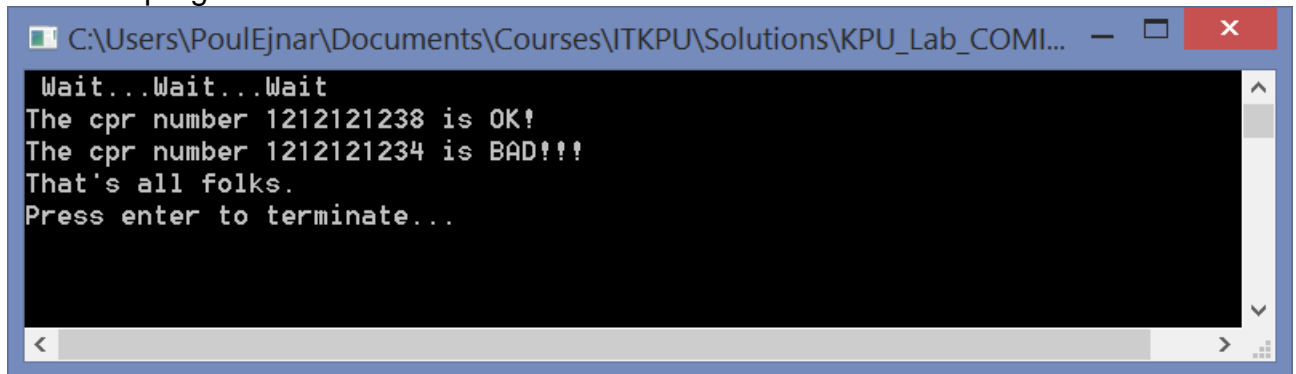


3. Under Project Settings – Build vælges “Register for COM interop”.



4. Tilføj attributten [ClassInterface(ClassInterfaceType.AutoDual)] til CprTest klassen.
5. Projektet rebuildes.
6. Laver et C++ console project med navnet CprComClient.

7. CPP klientprogrammet er nødt til at bruge smartpointers, da vi ingen header-fil har. Så start med et import-statement, og oversæt så programmet.
8. Herefter kan man åbne filerne *.tlh og *.tli for at se typenavne. Bemærk, at TlbExp sætter en '_' foran klassenavnet, når den skal lave et navn til interfacet.
9. Kørsel af program



```
C:\Users\PoulEjnar\Documents\Courses\ITKPU\Solutions\KPU_Lab_COMI...  
Wait...Wait...Wait  
The cpr number 1212121238 is OK!  
The cpr number 1212121234 is BAD!!!  
That's all folks.  
Press enter to terminate...
```

Alternativt bedre løsningsforslag:

En meget bedre Løsning kan ses i projekterne: CprBetterComServer og CprBetterComClient

Den væsentligste forskel er, at her angives eksplicit hvordan det eksporterede interface skal se ud. Herved kan man efterfølgende ændre i serverens implementering uden at ødelægge kompatibiliteten til de eksisterende klienter - forudsat at der ikke ændres i det eksisterende interface.

Delopgave 3

1. Laver et nyt C++ ATL projekt med navnet PizVizServer i en separat solution, og tilføjer 2 simple ATL objects: PizzaManager og OrderCalculator. Definerer deres grænseflader (som vist i ZIP-filen) og kopierer forretningslogikken (impl. Koden) fra det downloadede eksempel. Bygger COM-serveren – som også installerer den, hvilket kræver at Visual Studio køres som administrator.
2. Laver et C++ console program med navnet PizVizCppClient til at teste PizVizServer (kopierer koden fra det downloadede eksempel).
3. Laver et nyt WPF projekt og opbygger brugergrænsefladen - dette gøres i sammen solution som Lab1, da dette projekt ikke virker, hvis Visual Studio køres som administrator.
4. Tilføjer en reference til COM serveren fra trin1 – den har navnet PizVizServerLib. I ObjectBrowser'en ser jeg hvilke navne jeg skal brug. LookUp af pizzanavne er let, udfordringen i denne opgave er CalcOrder! COM-serveren forventer et SAFEARRAY, som ikke findes i C#. Den automatiske

marshal'er i CLR'en kan hjælpe os, men den kan kun marshal'e fra System.Array typen – så det kræver en lille omvej i koden.