

Class Library OOP 2.HF

It & Data, Odense

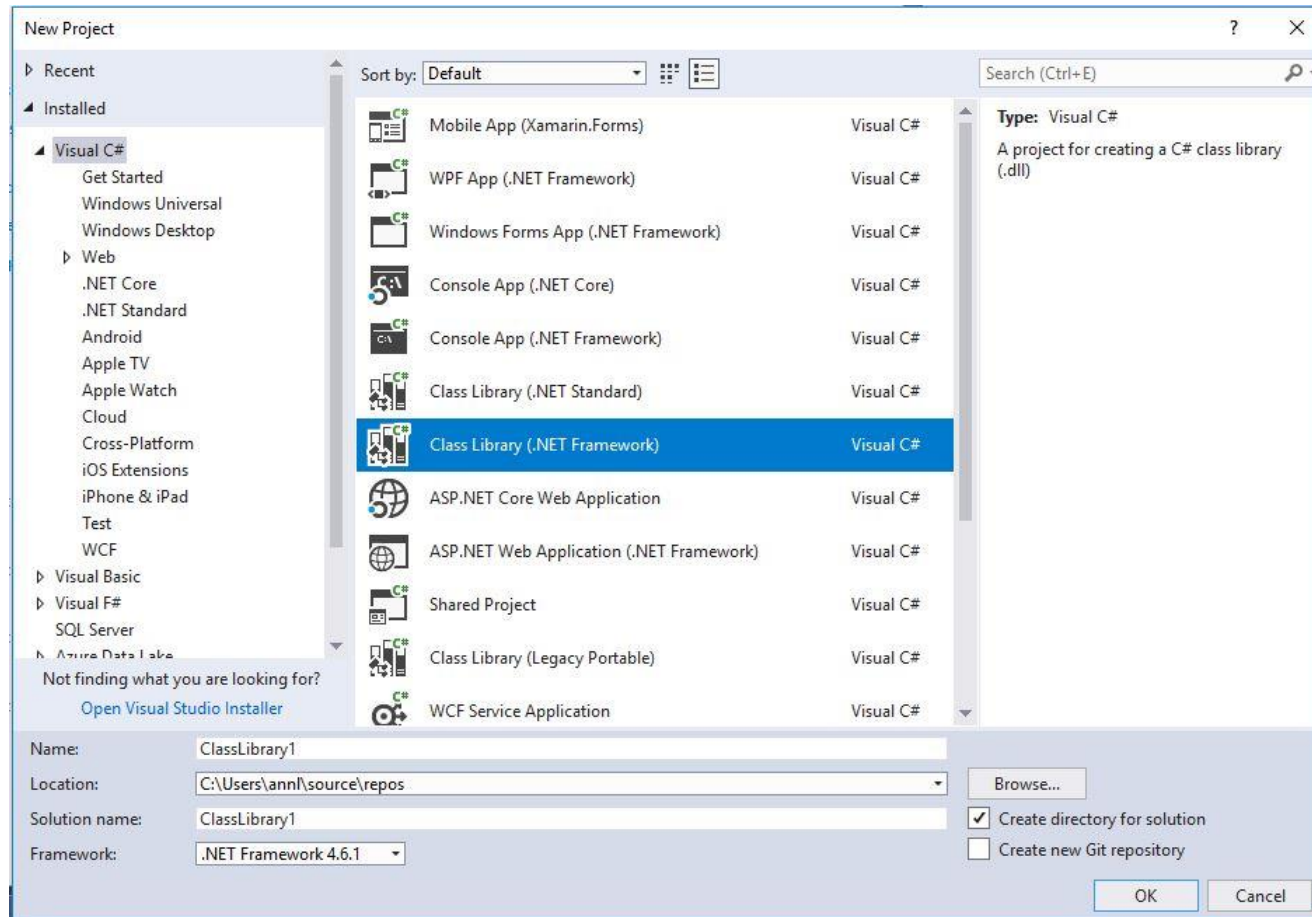
Class Library (dll)

- Hvad står det for?
 - DLL = Dynamic Link Library
- Hvad er ideen bag?
 - Genbrug af den samme funktionalitet/kode i flere forskellige projekter (Konsol, Windows Form, WPF, Web)
- Hvordan virker det?
 - En dll er en eksekverbar fil i lighed med en exe-fil, bort set fra en dll ikke kan køres på egen hånd
- Hvordan bruges den så?
 - Reference/import i et projekt
 - Include og derefter kan der instantieres objekter som normalt.
- Husk dll kan kreeres både som debug og release udgaver
 - **I et endeligt projekt refereres ALTID til release udgaven!**
 - Debug er kun til udviklingsbrug
- Råd: Test eventuelt koden i et almindeligt projekt, før den bygges ind i biblioteket.

<https://www.c-sharpcorner.com/UploadFile/1e050f/creating-and-using-dll-class-library-in-C-Sharp/>

Class Library (dll)

- Hvordan oprettes det?



- Et godt råd: Vælg et sigende navn, der antyder lidt om funktionaliteten.

Class Library - ideer

- I projekter med terningspil kunne et CL indeholde:
 - Terning, almindelig, snyd og valgfri antal øjne
 - Spillere, navn, score
 - Setup
- I projekter med handel:
 - Forskellige beregninger med moms (beskatning)
 - Valuta kurser
 - Fortjeneste / rabatter
- Formel samling
 - Beregning af areal, omkreds, volumen
 - Konvertering mellem enheder (f.eks. °C til °F og omvendt)
- Alment
 - Bruger oplysninger
 - Adgang/begrænsninger

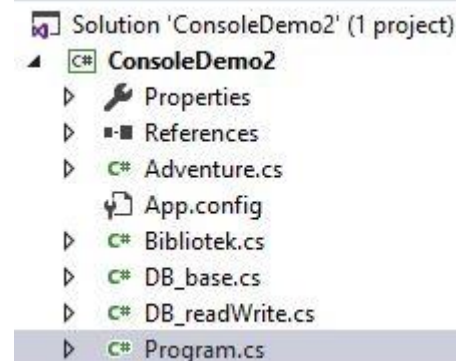
Det er kort sagt kun fantasien, der sætter grænser for, hvad et CL kan bruges til.

Class Library – Et eksempel

1. Formål at oprette en forbindelse til en MS SQL server
2. Det endelige bibliotek skal kunne anvendes til alle databaser på serveren
3. Mulighed for at forbinde via en server bruger
4. Mulighed for en standard forbindelse
5. Biblioteket skal kunne anvendes uanset projekt typen (Konsol, WinForm, WPF osv.)
6. Koden, der bruger biblioteket, har mulighed for at udbygge funktionaliteten (arv)

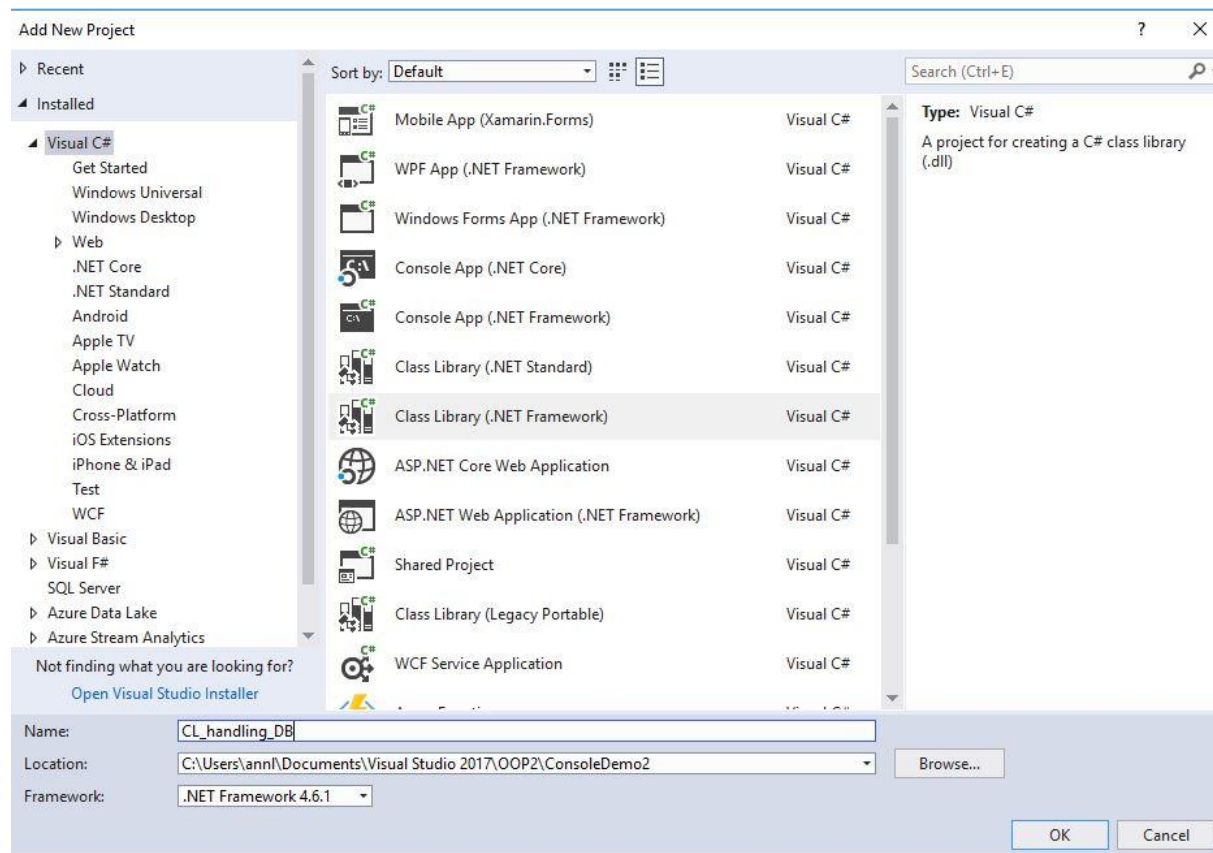
Class Library – et eksempel

- Dette projekt (en konsol applikation) består af fire klasser og en program klasse.
- DB_base er grundstammen, der opretter forbindelse til en MS SQL server.
- DB_readWrite arver forbindelsen fra DB_base og bygger videre med metoder, der kan læse og skrive i den valgte database.
- Bibliotek og Adventure klasser arver hver for sig fra DB_readWrite klassen. De er eksempler på to vidt forskellige databaser, der begge kan tilgås via de to grundklasser.
- I en demonstration i dag vises, hvordan de to klasser lægges over i et CL, der kan tilgås fra alle typer projekter.



Class Library - trin 1: Opret projekt

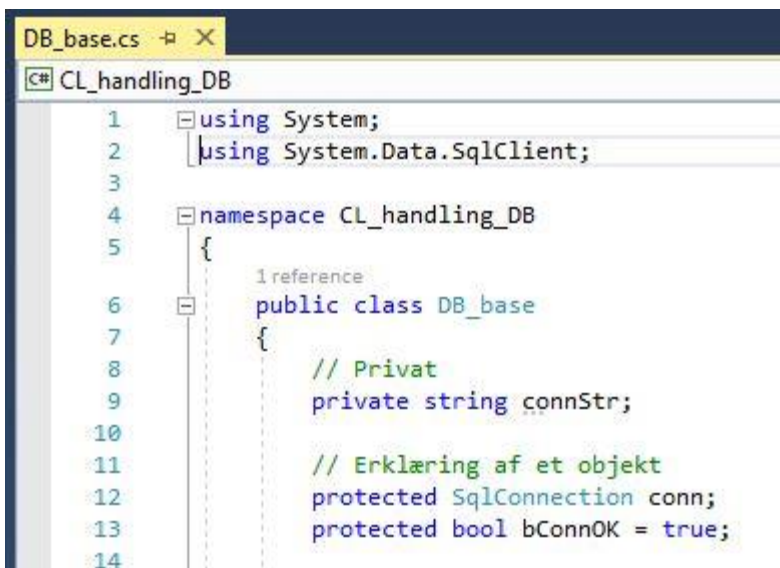
- Tilføj et nyt projekt (Add New Project) af typen 'Class Library (.NET Framework)'



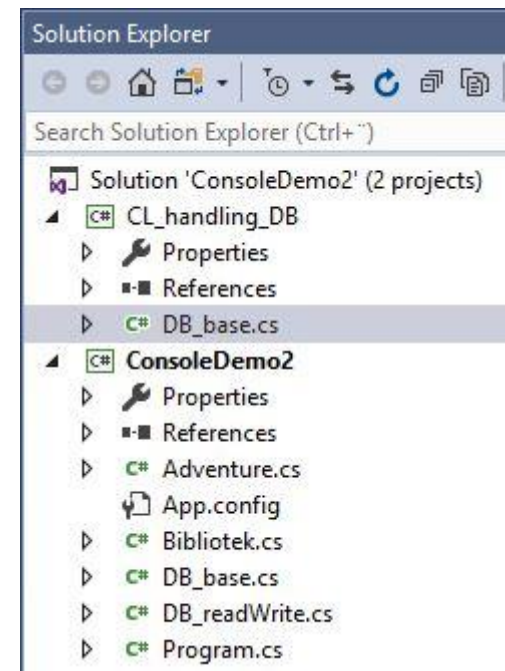
- Tænk over navnet – det skal bruges igen og igen, så et beskrivende navn er et klogt træk.

Class Library - trin 2:

- Omdøb default klassen 'Class1' til et navn, der beskriver **din** klasse bedst muligt. I dette eksempel bliver det til DB_base (som klasse fra det første projekt hed).
- Kopier koden fra den originale klasse over i biblioteks klassen.
- Husk at tilføje eventuelle referencer (using) til klassen – i dette tilfælde 'System.Data.SqlClient'



```
DB_base.cs
[CL_handling_DB]
1 using System;
2 using System.Data.SqlClient;
3
4 namespace CL_handling_DB
5 {
6     1reference
7     public class DB_base
8     {
9         // Privat
10        private string connStr;
11
12        // Erklæring af et objekt
13        protected SqlConnection conn;
14        protected bool bConnOK = true;
```

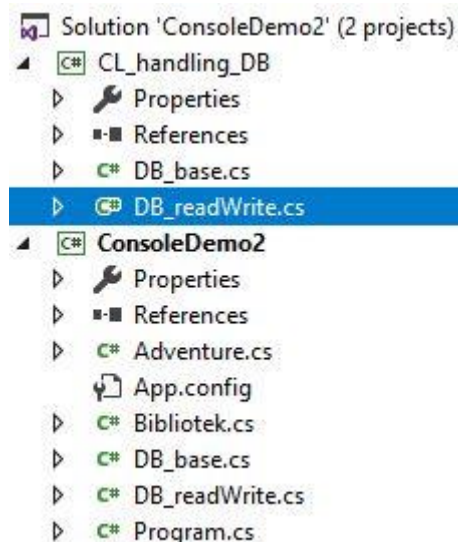


Class Library - trin 3:

- I dette tilfælde tilføjes en ny klasse 'DB_readWrite', der arver fra grund klassen 'DB_base' – hvordan det er i dit Class Library er op til dig. Men det er en god ide at oprette mere end en klasse.
- Igen i dette eksempel kopieres den eksisterende kode over i den nye klasse.
- Husk at tilføje en 'Access modifier' til klassen alt efter behov (Public, Private, Protected eller Internal).

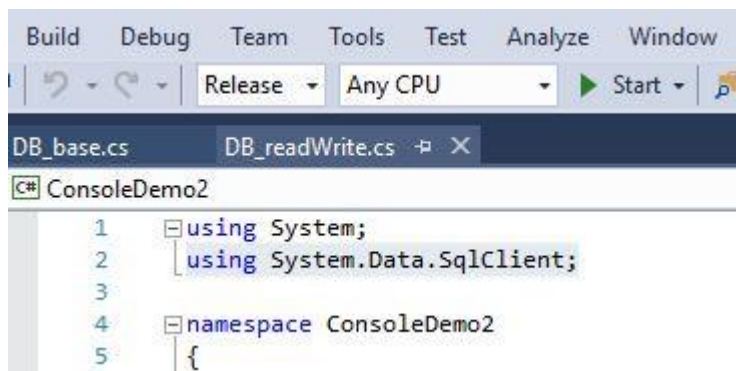


```
1 using System;
2 using System.Data.SqlClient;
3
4 namespace ConsoleDemo2
5 {
6     3 references
7     public class DB_readWrite : DB_base
8     {
9         2 references
10        public DB_readWrite(string dbName) : base(dbName)
11        {
12            cmd = new SqlCommand();
13            cmd.Connection = conn;
14        }
15        SqlCommand cmd;
```



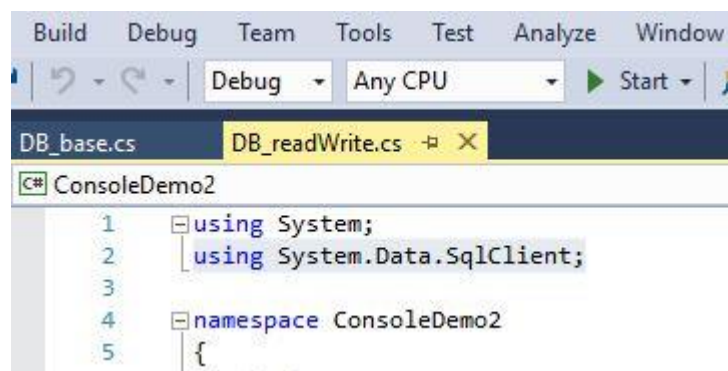
Class Library - trin 4

- Alt gemmes, og projektet kompileres (Build).
- HUSK at en dll i Debug mode er kun til test.
- Brug ALTID en release dll i dit endelige projekt.



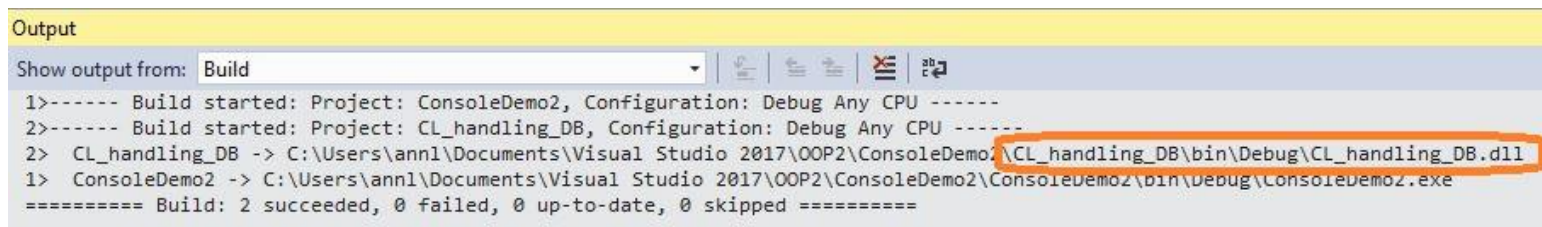
Visual Studio interface showing the Release configuration for the ConsoleDemo2 project. The Build menu is open, and the Release option is selected. The output window shows the build process for the Release configuration.

```
1 using System;
2 using System.Data.SqlClient;
3
4 namespace ConsoleDemo2
5 {
```



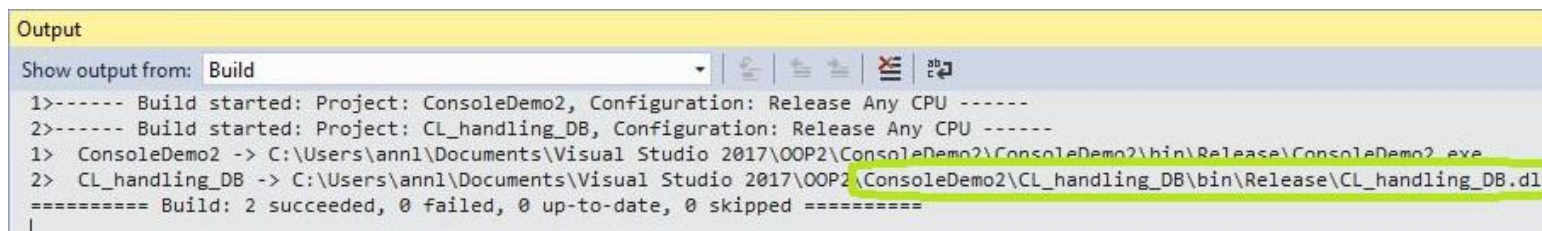
Visual Studio interface showing the Debug configuration for the ConsoleDemo2 project. The Build menu is open, and the Debug option is selected. The output window shows the build process for the Debug configuration.

```
1 using System;
2 using System.Data.SqlClient;
3
4 namespace ConsoleDemo2
5 {
```



Output window showing the build process for the Release configuration. The output shows the build of the ConsoleDemo2 project and the CL_handling_DB project, resulting in the CL_handling_DB.dll file.

```
1>----- Build started: Project: ConsoleDemo2, Configuration: Debug Any CPU -----
2>----- Build started: Project: CL_handling_DB, Configuration: Debug Any CPU -----
2> CL_handling_DB -> C:\Users\ann1\Documents\Visual Studio 2017\OOP2\ConsoleDemo2\CL_handling_DB\bin\Debug\CL_handling_DB.dll
1> ConsoleDemo2 -> C:\Users\ann1\Documents\Visual Studio 2017\OOP2\ConsoleDemo2\ConsoleDemo2\bin\Debug\ConsoleDemo2.exe
===== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

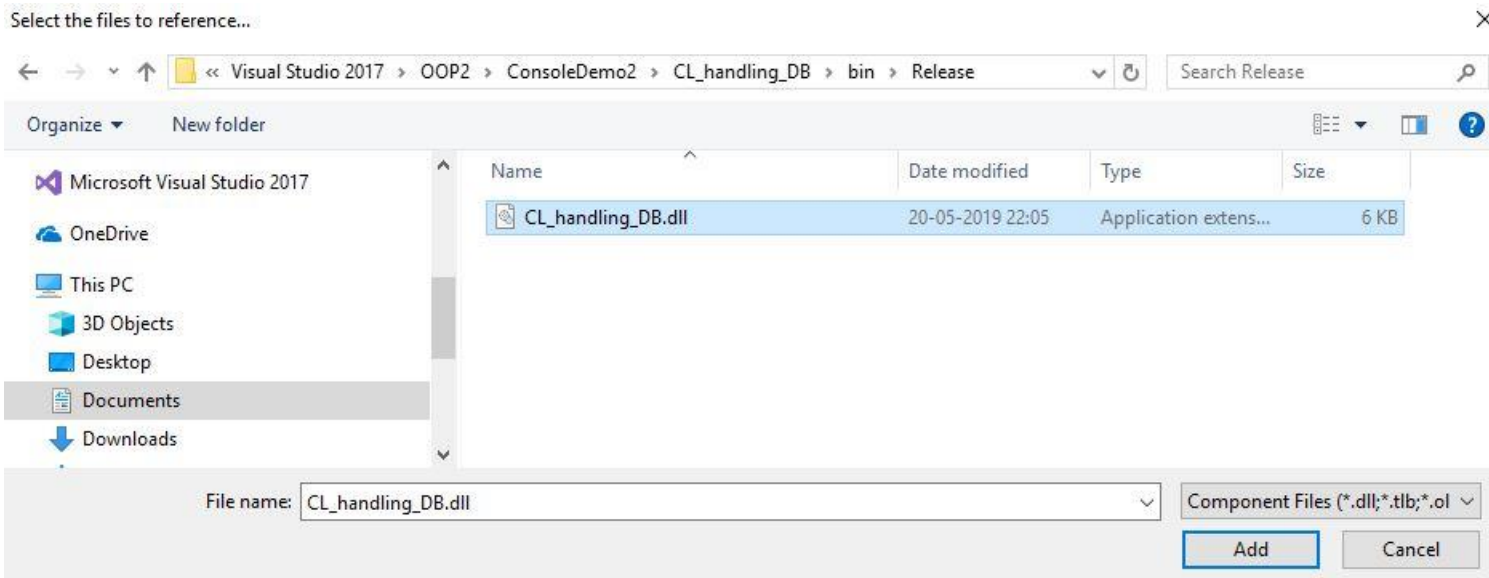
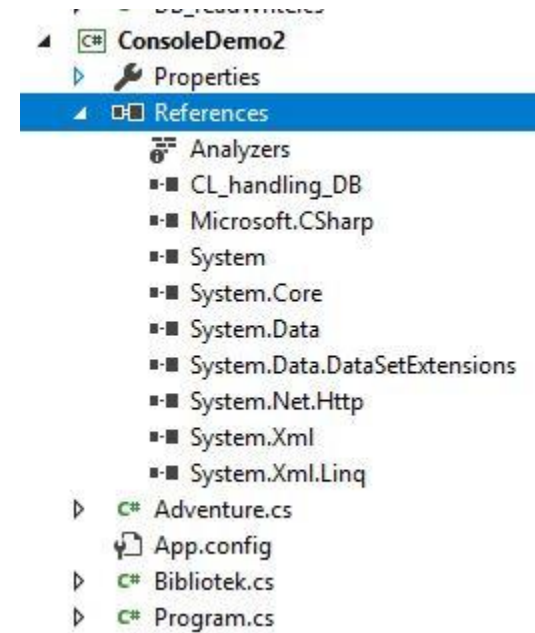


Output window showing the build process for the Release configuration. The output shows the build of the ConsoleDemo2 project and the CL_handling_DB project, resulting in the CL_handling_DB.dll file.

```
1>----- Build started: Project: ConsoleDemo2, Configuration: Release Any CPU -----
2>----- Build started: Project: CL_handling_DB, Configuration: Release Any CPU -----
1> ConsoleDemo2 -> C:\Users\ann1\Documents\Visual Studio 2017\OOP2\ConsoleDemo2\bin\Release\ConsoleDemo2.exe
2> CL_handling_DB -> C:\Users\ann1\Documents\Visual Studio 2017\OOP2\ConsoleDemo2\CL_handling_DB\bin\Release\CL_handling_DB.dll
===== Build: 2 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

Class Library Test

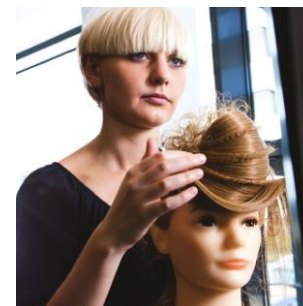
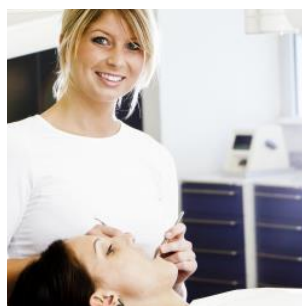
- I det originale projekt slettes de to original klasser.
- I stedet for tilføjes det nye Class Library til referencerne.
- I de to tilbageværende klasser tilføjes using CL_handling_DB
- Nu er projektet klar til brug – med et indbygget Class Library



Class Library - konklusion

- Ideen bag et class library er selvfølgelig ikke begrænset til et projekt.
- Den færdige dll-fil kan anvendes i alle nye projekter, hvor der er behov for den type funktionalitet, den står for.
- Det kræves naturligvis, at den er designet godt nok.
 - Koden skal være uafhængig af kommandoer, der knytter sig til bestemte projekt typer, som f.eks. `Console.WriteLine` (der kun anvendes i et konsol projekt)

Slut på Class Library OOP



Campus M – Vi uddanner Danmarks dygtigste