# WPF Simple GUI

|  |  |
| --- | --- |
| Læringsmål | Du kan:   * **1Pf1**: anvende centrale metoder til at specificere og konstruere algoritmer [...] * **1Pf2**: anvende centrale faciliteter i programmeringssproget til realisering af algoritmer, designmønstre, abstrakte datatyper, datastrukturer, designmodeller anvende centrale faciliteter i programmeringssproget til realisering af algoritmer, designmønstre, abstrakte datatyper, datastrukturer, designmodeller og brugergrænseflader |
| Forventet læringsudbytte [SOLO] | Programmeringssporet:   * [Unistrukturel] Du kan identificere enkelte kendetegn ved en **GUI-applikation** – WPF og XAML |
| Din forberedelse | Programmeringssporet:  (*Selvom nedenstående videoer og hjemmesider har nogle år på bagen og gennemgår WPF til .NET Framework og viser tidligere versioner af Visual Studio, er de stadig dækkende for WPF til .NET 5, selvom der kan være små forskelle*)   * [Overview of Windows Presentation Foundation](https://www.linkedin.com/learning/windows-presentation-foundation-1-build-dramatic-desktop-applications/overview-of-windows-presentation-foundation) (video: 5:35) * [Overview of XAML](https://www.linkedin.com/learning/windows-presentation-foundation-1-build-dramatic-desktop-applications/overview-of-xaml) (video: 3:34) * [Use the XAML editor](https://www.linkedin.com/learning/windows-presentation-foundation-1-build-dramatic-desktop-applications/use-the-xaml-editor) (video: 9:50) * Læs nedenstående sektioner fra [The complete WPF tutorial](https://www.wpf-tutorial.com/): *(Artiklerne er oversat til dansk; dette er angivet øverst i hver artikel)*   + [What is WPF?](https://www.wpf-tutorial.com/about-wpf/what-is-wpf/)   + [Hello WPF](https://www.wpf-tutorial.com/getting-started/hello-wpf/)   + XAML (skimlæs detaljerne omkring events)     - [What is XAML?](https://www.wpf-tutorial.com/xaml/what-is-xaml/)     - [Basic XAML](https://www.wpf-tutorial.com/xaml/basic-xaml/)     - [Events in XAML](https://www.wpf-tutorial.com/xaml/events-in-xaml/) (lige nu skal du bare bruge nogle få events og ikke forstå dem i dybden; det kommer vi til inden længe)   + A WPF application:     - [A WPF Application – Introduction](https://www.wpf-tutorial.com/wpf-application/introduction/)     - [The Window](https://www.wpf-tutorial.com/wpf-application/the-window/)     - [Working with App.xaml](https://www.wpf-tutorial.com/wpf-application/working-with-app-xaml/)     - De resterende afsnit 4-7 kan skimlæses.   + Basale UI-kontroller:     - Læs om [Label](https://www.wpf-tutorial.com/basic-controls/the-label-control/), [TextBox](https://www.wpf-tutorial.com/basic-controls/the-textbox-control/) og [Button](https://www.wpf-tutorial.com/basic-controls/the-button-control/) * Gense:   + [C# Value & Reference types](https://www.youtube.com/watch?v=eehaSv4vofA) (video: 3:26) |

I denne opgave skal du lave din første WPF-applikation, hvilket vil sige en C#-applikation med en grafisk brugergrænseflade (GUI). Du får indblik i opbygningen af en WPF-applikation, i XAML, hvor GUI’ens opbygning beskrives i XML-format, i code-behind, hvor den UI-relaterede C#-logik befinder sig, samt nogle få basale UI-kontroller til at komme i gang med. Gennem to øvelser, hvoraf den ene er en bonusøvelse, skal du skabe en dynamisk interaktion med brugeren via GUI’en.

# Dagens ord:

If we want users to like our software, we should design it   
to behave like a likeable person: respectful, generous, and helpful.”  
―  
(Alan Cooper, Software Designer and Programmer)

# Øvelse 1: Terminologi

Del teamet op i to mindre grupper og brug **Ordet rundt** til at reflektere over begreberne ”Value type”, ”Reference type”, ”WPF”, ”XAML”, ”Code-behind”, ”TextBox”, ”Button” og ”Label”. Sørg for at alle får mulighed for at tale.

Tidsramme: 15 minutter

**Value type**: Simple data types, e.g. int, double, float, bool, char, that stores the value in the variable, as opposed to **reference type,** e.g. strings, classes, interfaces, where the variable stores a location in the memory that refers to the type. So two variable that refer to the same value is either duplicated if value type or referencing the same value on the memory. Reference types are stored in heap memory, value types are stored in the stack memory.??

**WPF** stands for Windows Presentation Foundation and is a UI framework to build Windows desktop applications.

**XAML** stands for eXtensible Application Markup Languge and is the (can be used) language used to design the UI layer of a dotnet application.

**Code-behind** is the logic that is behind the the UI, what happens when different events occur in the UI, i.e. the business logic.

**TextBox** is a user text input box. A control.

**Button** is a control that can be clicked.

**Label** is usually a label for another control, e.g. a textbox, it is like a TextBlock, but can contain an image and can be accessed with the alt+letter keybind that is common for windows application, e.g. word.

**Benyt parprogrammering til alle følgende øvelser.**

# Øvelse 2: WPF GUI

Du skal nu begynde at arbejde med grafiske brugergrænseflader (GUI) i stedet for konsolapplikationer. Dette introducerer mange flere muligheder for at lave moderne brugergrænseflader med langt større kompleksitet. I denne opgave starter du med at se på de helt basale UI-komponenter til en GUI.

## Øvelse 2.1: Indledende opsætning

Udfør følgende:

* Opret et nyt Visual Studio projekt med navnet ”WPFSimpleGUI”, hvor du vælger **WPF Application** som projektskabelon
* Gør dig nu bekendt med projektets filer i Solution Explorer:  
  Et billede, der indeholder tekst

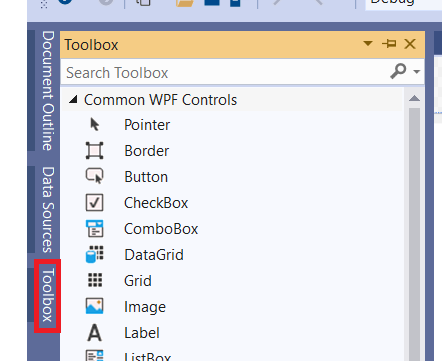
  Automatisk genereret beskrivelse
  + Med specielt henblik på:
    - ”MainWindow.xaml”, som indeholder både en XML-repræsentation af den grafiske UI (XAML) og en grafisk designvisning
    - ”MainWindow.xaml.cs”, som er **code-behind** filen bag den grafiske UI

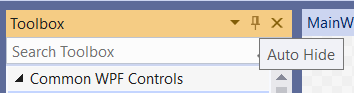
Det er disse to filer, du skal arbejde med i denne øvelse.

## Øvelse 2.2: Placér UI-kontroller på GUI’en

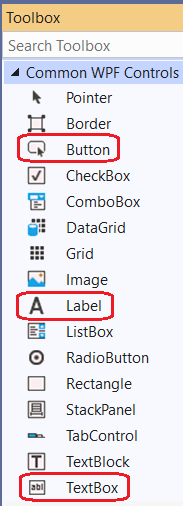
Når du opbygger GUI’en, skal du til denne øvelse anvende tre typer UI-kontrollere: *Label*, *TextBox* og *Button*

Udfør følgende:

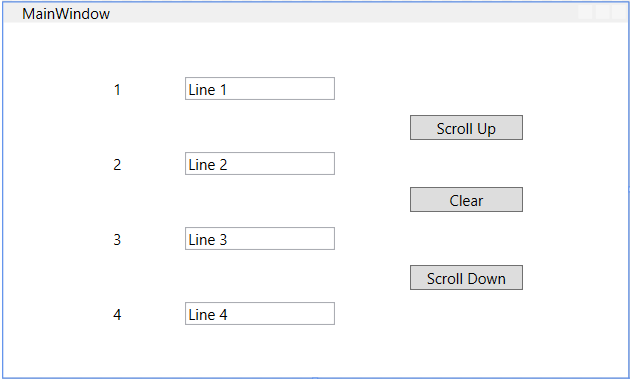
* Åbn MainWindow.xaml filen, så du kan se brugergrænsefladen, både grafisk og i XAML-format
* I XAML-vinduet skal du se under ”Window” og ændre *Height* til at være 300 og *Width* til at være 500
* Åbn Toolbox-vinduet (View -> Toolbox)
  + Husk at benytte “Auto Hide” for at fastgøre Toolbox-vinduet i menuen til venstre:



* I Toolbox-vinduet vil der være en liste af forskellige UI-kontroller, som du kan vælge fra, f.eks. fra sektionen ’Common WPF Controls’:



* Træk de relevante kontrollere over i MainWindow.xaml, så den indeholder fire *Labels*, fire *TextBoxes* og tre *Buttons* med følgende layout og indhold:

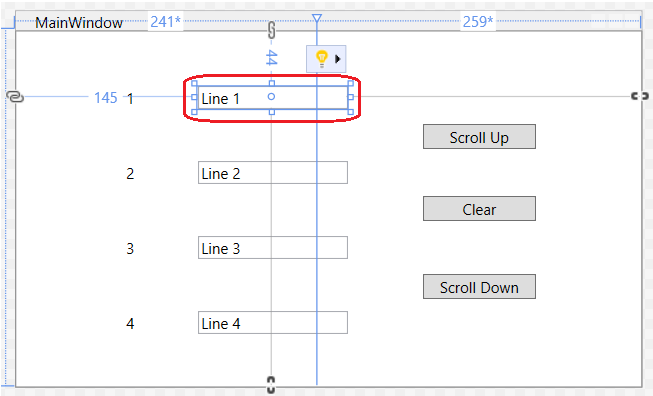
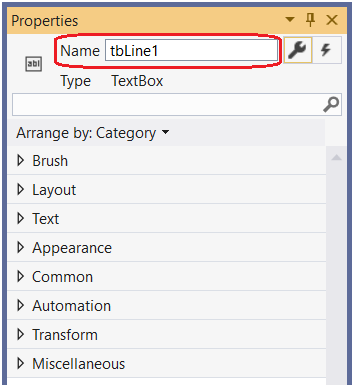


Med denne GUI ønskes der følgende funktionalitet:

* Ved klik på ”Scroll Up” flyttes informationen i tekstboksene ”en linje op”, forstået på den måde, at indholdet i tekstboks 4 flyttes til tekstboks 3, indholdet i tekstboks 3 flyttes til tekstboks 2, osv. Bemærk, at indholdet tekstboks 1 flyttes til tekstboks 4
* Ved klik på ”Clear” ryddes indholdet i alle tekstboksene
* Ved klik på ”Scroll Down” flyttes informationen ”en linje ned”, dvs. det modsatte af ”Scroll Up”

***Hint 1:*** *En Button-kontrol har et click event, så når du dobbeltklikker på en Button-kontrol i design-mode, så oprettes automatisk en code-behind event-handler, du kan kode i. Det er denne kode, som afvikles, når du klikker på knappen (button) i et kørende program. Du kan selvfølgelig også debugge denne kode.*

***Hint 2:*** *I code-behind filen kan du referere til de enkelte UI-komponenter på GUI’en via den navngivning, du giver hvert element, og dermed styre dem via koden. Navnet sættes i Name-property i Properties-vinduet (View --> Properties Window).*

Udfør følgende:

* Implementér ovenstående funktionalitet i code-behind filen (MainWindow.xaml.cs)
* Kør programmet og afprøv, om det virker efter hensigten. Tilret, hvis ikke
  + ***Hint:*** Value types vs reference types

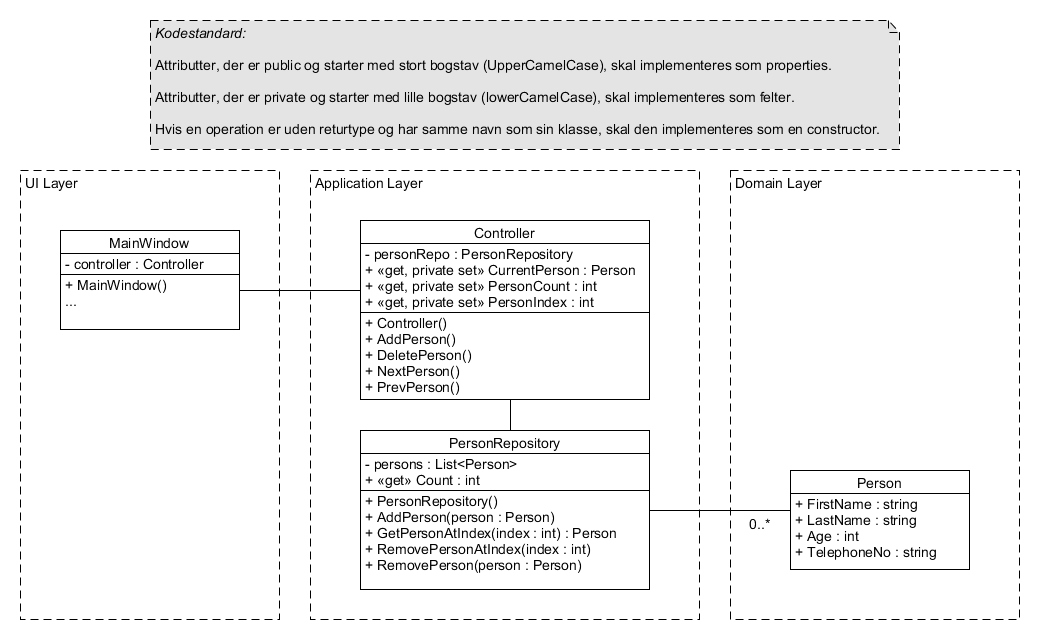
# Øvelse 3: Interaktiv GUI

Du skal rette i en eksisterende WPF-løsning med en mere komplet arkitektur, så den fungerer som ønsket.

## Øvelse 3.1: Klargøring

Udfør følgende:

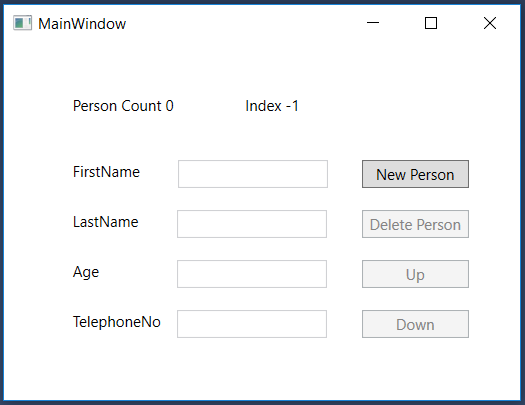
* Benyt ’git pull’ i git materiale-folderen og sikr dig, at du kan se en ny fil: Ex24-WPFInteractiveGUI.zip
* Pak zip-filen ud, og åben løsningen ”WPFInteractiveGUI”. Den indeholder et projekt af typen ”WPF Application” med de 3 klasser *Controller*, *PersonRepository* og *Person* som vist i følgende lagdelte klassediagram:



Derudover indeholder projektet selve GUI’en (MainWindow.xaml) og dens code-behind fil (MainWindow.xaml.cs). Det er udelukkende disse to filer, du skal ændre i opgaven. Dvs. du må i denne opgave IKKE ændre koden til Controller, PersonRepository og Person.

Udfør følgende:

* Inspicér først nøje alle klasser i den udleverede kode
* Træk komponenter over i MainWindow.xaml fra jeres ToolBox, så GUI’en kommer til at se ud som følger (Den skal indeholde kun Labels, Textboxes og Buttons):



Sørg for, at data for den valgte person i repositoriet vises i GUI’en med følgende funktionelle krav:

* Ved programstart er repositoriet tomt (som vist foroven)
* Hvis repositoriet er tomt (Person Count viser 0, og Index viser -1) skal følgende altid gælde:
  + Alle fire tekstbokse er uden indhold og er deaktiverede (IsEnabled = false)
  + Knapperne ”Delete Person”, ”Up” og ”Down” er ligeledes deaktiverede (IsEnabled = false)
  + Knappen ”New Person” er aktiveret (IsEnabled = true)

Alle UI-kontroller har property’en IsEnabled, en boolsk egenskab der indikerer, hvorvidt UI-kontrollen er aktiveret i brugergrænsefladen eller ej.

* Det er yderligere et krav i jeres løsning, at GUI’ens code-behind, dvs. MainWindow.xaml.cs, KUN må gøre brug af Controlleren (uden nogen reference til PersonRepository eller Person) til at oprette en ny person, slette udvalgt person og via Controller’ens oplysninger sørge for, at vinduet altid viser data for den valgte person i PersonRepository (med mindre repositoriet er tomt)

Udfør følgende:

* Ved klik på ”New Person”, kaldes AddPerson i Controlleren, så den nye person (uden data) indsættes i repositoriet og gøres til den valgte person (CurrentPerson) i Controller’en  
  Den ændring skal afspejles i vinduet, så alle GUI-komponenter er aktive
* Ved enhver ændring af indholdet i en af tekstboksene, skal data for den valgte person opdateres i Controlleren (via CurrentPerson)
* Ved klik på ”Delete Person”, kaldes DeletePerson i Controlleren. Dette skaber en ændring i CurrentPerson, der skal afspejles i vinduet. Hvis repositoriet er tomt, da husk kravene foroven
* Ved klik på ”Up” eller ”Down” kaldes tilsvarende metoder i Controlleren, hvilket indebærer en ny valgt person (CurrentPerson). Denne ændring skal afspejles i vinduet

***Hint:*** *Tekstboksen har et event, der aktiveres hver gang, tekstindholdet ændres.*

HUSK at debugge, hvis det ikke virker som forventet!

# Øvelse 4: Vidensdeling

Benyt tre-til-te CL-strukturen til vidensdeling.