# WPF Simple GUI og Tusindfryd

# 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”, ”Label” og ”IsEnabled property (for en WPF control)”. Sørg for at alle får mulighed for at tale.

Tidsramme: 15 minutter

**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

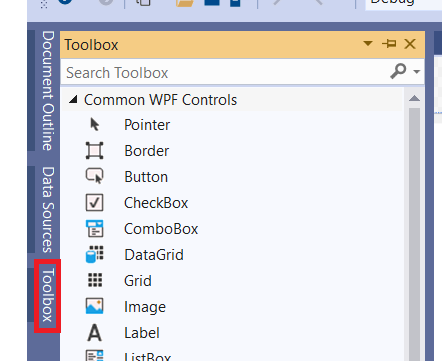
  Automatisk genereret beskrivelse
  + Med specielt henblik på:
    - ”MainWindow.xaml”, som indeholder både en XML-repræsentation af den grafiske UI (XAML) og en grafisk designvisning
    - ”MainWindow.xaml.cs”, som er **code-behind** filen bag den grafiske UI

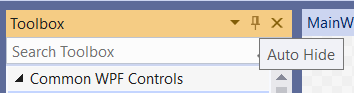
Det er disse to filer, du skal arbejde med i denne øvelse.

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

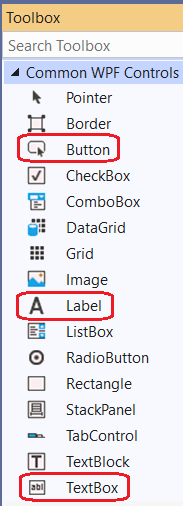
Når du opbygger GUI’en, skal du til denne øvelse anvende tre typer UI-kontrollere: *Label*, *TextBox* og *Button*

Udfør følgende:

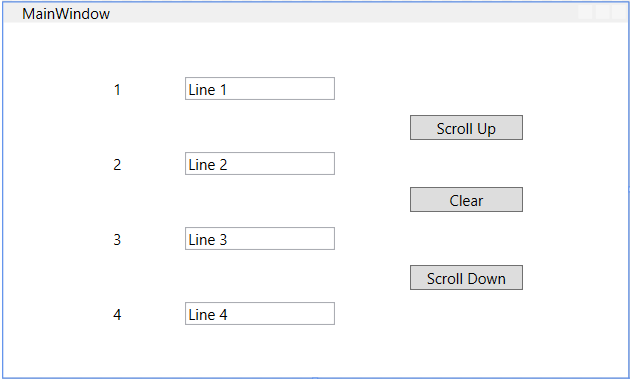
* Åbn MainWindow.xaml filen, så du kan se brugergrænsefladen, både grafisk og i XAML-format
* I XAML-vinduet skal du se under ”Window” og ændre *Height* til at være 300 og *Width* til at være 500
* Åbn Toolbox-vinduet (View -> Toolbox)
  + Husk at benytte “Auto Hide” for at fastgøre Toolbox-vinduet i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

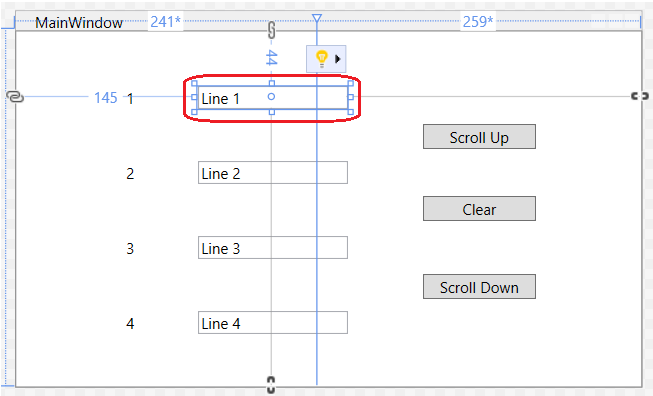
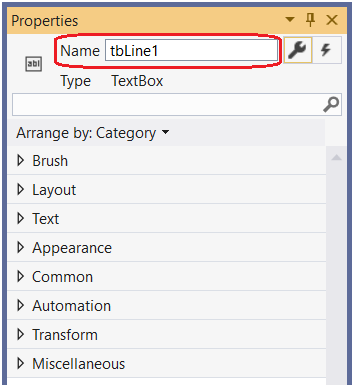


Med denne GUI ønskes der følgende funktionalitet:

* 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

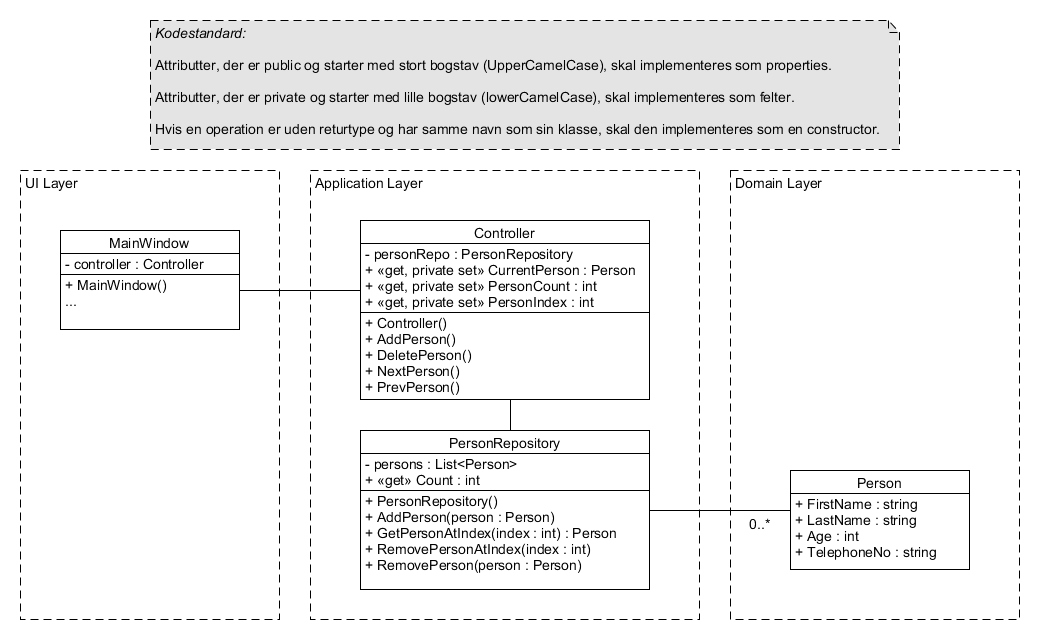
Udfør følgende:

* Benyt ’git pull’ i git materiale-folderen og sikr dig, at du kan se en ny fil: Uge18\_1-WPFInteractiveGUI.zip
* Pak zip-filen ud, og åbn løsningen ”WPFInteractiveGUI”. Hvis du bliver bedt om at stole på løsningen via en Trust-besked:

Et billede, der indeholder tekst, Font/skrifttype, linje/række, skærmbillede

Automatisk genereret beskrivelse  
så vælg ’Trust and continue’ som vist foroven.

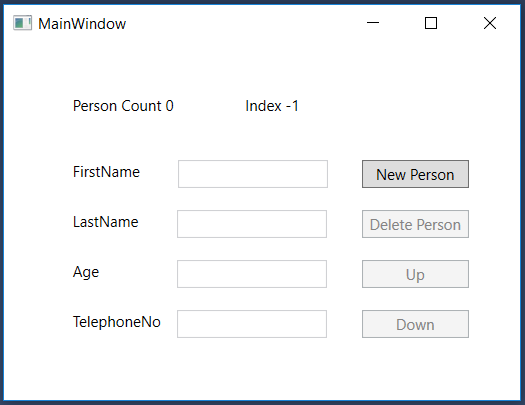
* Løsningen indeholder et projekt af typen ”WPF Application” med de 3 klasser *Controller*, *PersonRepository* og *Person* som vist i følgende lagdelte klassediagram:



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’-label viser 0, og ’Index’-label 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 boolesk 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)
  + ***Hint:*** *Tekstboksen har et event, der aktiveres hver gang, tekstindholdet ændres*
* 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

HUSK at debugge, hvis det ikke virker som forventet!

# Øvelse 4: Terminologi

Del teamet op i to mindre grupper og brug **Ordet rundt** til at reflektere over begreberne ”Message boxes”, ”Common dialog boxes”, ”Custom dialog boxes”, ”WPF TextBlock *control*” og ”WPF Image control”. Sørg for at alle får mulighed for at tale.

*Tidsramme: 15 minutter*

Overvej også, hvad forskellen er på en WPF TextBlock control og en WPF Label control, inden du går videre.

**Benyt parprogrammering til alle følgende øvelser.**

# Øvelse 5: Tusindfryd

I systemudvikling fik du udleveret artefakterne for Tusindfryd casen. Nu kigger du nærmere på etablerNyBlomstersort-operationskontrakten i TusindFryd-casen. Denne operationskontrakt definerer forskellige detaljer for en tilsvarende system­operation i SSD’en. Kigger man nærmere på operationskontrakten, er der flere oplysninger, der er nødvendige for at kunne oprette en ny blomstersort:

* Navn (tekststreng med navn)
* Billede (binær data, men for nu en tekststreng, som angiver stien til billedet)
* Produktionstid (heltal)
* Halveringstid (heltal)
* Størrelse (heltal)

Disse oplysninger skal alle hentes fra brugeren.

I det følgende skal du udvikle en WPF-brugergrænseflade, som henter disse oplysninger via en dialogboks. Dvs. du skal både have:

1. Et hovedvindue (MainWindow) i din løsning
2. En dialogboks (et ekstra vindue) til at indhente oplysninger for en ny blomstersort
3. Kode til at åbne din dialogboks fra hovedvinduet
4. Kode til at gemme de indtastede blomstersort-oplysninger fra dialogboksen i en liste-datastruktur i hovedvinduet, f.eks. List<T> (lige nu behøver du ikke at overveje et repository)
5. UI-kontrol og kode til at vise tilføjelsen af en blomstersort i en passende UI-kontrol i hovedvindue, f.eks. en TextBlock, der viser alle blomstersorter i liste-datastrukturen

## Øvelse 5.1: Klargøring

Udfør følgende:

* Opret en ny WPF-applikation ”TusindfrydWPF” (husk at benytte projektskabelonen ”WPF Application”)
* Tilpas Height og Width i Window-elementet i XAML-filen for MainWindow
  + Til for eksempel henholdsvis 250 og 300
* Et billede, der indeholder tekst, skærmbillede, display/skærm/fremvisning, software

  Automatisk genereret beskrivelseTilføj følgende UI-kontroller til MainWindow:
  + Indsæt øverst i hovedvinduet en Label kontrol med teksten ”Blomstersorter:”
  + Indsæt herunder en TextBlock. Prøv om du kan sætte baggrunden (Background) på TextBlock’en til lysegrå (LightGray), så den er synlig. Du vender tilbage til indholdet om lidt
  + Indsæt nederst i hovedvinduet en enkel knap (Button) med titlen ”Opret Blomstersort”.  
    Knappen vender du tilbage til om lidt  
    Det skulle gerne nogenlunde ende med billedet til højre ->
* Tilføj en C#-klasse til projektet med navnet ”FlowerSort” til at repræsentere en blomstersort  
  ud fra nedenstående software designklasse og ordliste:

|  |  |  |
| --- | --- | --- |
| Designklasse | Dansk | Engelsk |
| Et billede, der indeholder tekst, Font/skrifttype, skærmbillede, hvid  Automatisk genereret beskrivelse | Blomstersort | FlowerSort |
| Navn | Name |
| Billede (sti) | PicturePath |
| Produktionstid | ProductionTime |
| Halveringstid | HalfLifeTime |
| Størrelse | Size |

* Opret til sidst en liste i MainWindow codebehind-filen til at gemme alle indtastede blomstersorter, f.eks som angivet med variablen *flowerSorts* forneden:



## Øvelse 5.2: Opret dialogboks til blomstersort

Næste trin er at oprette selve dialogboksen til at indhente oplysninger for en ny blomstersort, hvor du skal lave et layout med Labels, Textbox’e og en enkelt Image UI-kontrol til at indtaste de nødvendige oplysninger samt klikke ’OK’ eller ’Fortryd’. Et eksempel på et GUI-design kunne være:

Et billede, der indeholder tekst, kat, skærmbillede, software

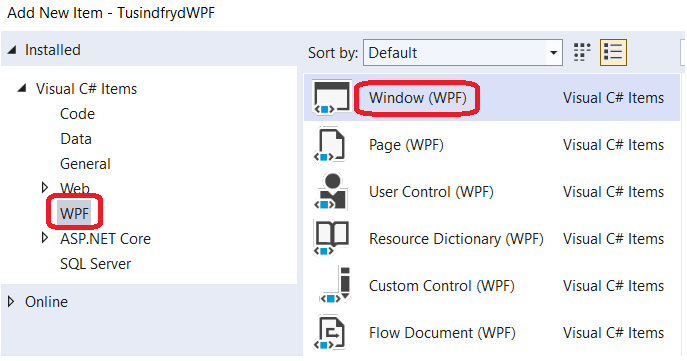
Automatisk genereret beskrivelse

Funktionelle krav:

* Alle felter skal være udfyldt for, at en gyldig blomstersort accepteres ved klik på OK-knappen
* Image UI-kontrollen skal vise det billede, som stien peger på (foroven vises bare et eksempel). Så snart TextBox’en for billedstien mister fokus (LostFocus-eventet) skal det angivne billede vises, hvis muligt. Hvis der opstår en fejl, så håndtér den med brug af exceptions (try-catch-finally).  
  Selve billedet skal ikke gemmes i FlowerSort-objektet, kun stien til billedet

Udfør følgende:

* Opret nu et helt nyt WPF-vindue, som skal bruges som dialogboks til at indhente blomstersort-oplysningerne:
  + Højreklik på ”TusindfrydWPF”-projektet i Solution Explorer og vælg ”Add” og derefter ”New Item …”. Herved skulle du gerne se nedenstående dialogboks:



* + Vælg ”WPF” under ”C# Items”, og derefter “Window (WPF)” som vist med rødt
  + Navngiv det nye vindue ”*CreateFlowerSortDialog*”, og tryk ”Add”-knappen
* Tilpas Height og Width under Window i XAML-filen for CreateFlowerSortDialog
  + Til for eksempel henholdsvis 400 og 400
* Tilføj alle de angivne UI-kontroller til dialogboksen
* Kod den nødvendige funktionalitet for dialogboksen, og afprøv  
  Vent til sidst med billedefunktionaliteten til alt det andet virker.  
  **Husk at debugge med breakpoints, hvis der opstår problemer.**

## Øvelse 5.3: Aktiver dialogboks fra hovedvindue og hent resultat (ny blomstersort)

Du har oprettet hovedvinduet og dialogboksen, men mangler stadig nogle få trin. Når brugeren klikker på ”Opret Blomstersort”-knappen i hovedvinduet, skal dialogboksen åbnes, så blomstersortoplysninger kan indtastes.

Udfør følgende:

* I Click-eventhandler for ”Opret Blomstersort”-knappen, opret en instans af dialogboksen, kald dens ShowDialog-metode, og håndter den returnerede booleske værdi. Find ud af, hvad returværdien betyder.
* Afprøv din kode, og tilret, hvis fejl. Husk debug og exception handling.

## Øvelse 5.4: Overfør resultat fra dialogboks (ny blomstersort)

Du kan nu åbne dialogboksen og indtaste oplysninger for en ny blomstersort, men vil selvfølgelig gerne kunne overføre disse oplysninger, så snart du trykker OK og returnerer til Click-eventhandler’en for ”Opret Blomstersort”-knappen.

Overvej:

* Hvordan får du fat i de 5 oplysninger fra dialogboksen?
* Implementér din løsning på problemet.  
    
  ***Hint:*** en public variabel i dialogboksens klasse kunne måske være vejen frem

## Øvelse 5.5: Opdatér liste af blomstersorter

Du har nu fået oplysninger fra dialogboksen tilbage i hovedvinduet, så det sidste, der mangler, er at opdatere listen af blomstersorter (flowerSorts) og genopfriske hovedvinduets TextBlock, så den viser en tekstlinje for hver blomstersort i listen.

* Implementér dette, og afprøv din applikation

## Øvelse 5.6: Styring af OK-knappen (Enable/Disable)

Du vil nu gerne lave en lidt mere smart styring af OK-knappen i dialogboksen, så knappen kun er aktiveret (Enabled), når alle felter er udfyldt, og ellers deaktiveret (Disabled).

Udfør følgende:

Tilret koden i dialogboksen, så denne funktionalitet opnås.  
  
***Hint:*** der er en eventhandler, som kaldes hver gang teksten i en TextBox ændres.

# Øvelse 6: Vidensdeling (kl. 14.30)

Vi mødes i generel til vidensdeling.