Opgave 1 WPF - Introduktion

Objekter og nedarvning

# Formål

Formålet med opgaven er at lære hvordan man grundlæggende anvender WPF til at lave en GUI som gør brugeren i stand til at løse en opgave. Du opnår grundlæggende forståelse for hvordan man udnytter de faciliteter Visual Studio stiller til rådighed, når det drejer sig om design af en GUI.

# Mål

Opgavens mål er

1. at kunne oprette et WPF-projekt
2. at kunne lave en GUI der opfylder kravet i opgaven
3. at kunne benytte standard elementer fra Tool Box
4. at kunne udvælge og benytte de rette eventhandlere
5. at kunne anvende metoder til udførelse af funktionalitet

# 1 Struktur – Grid med Row & Colum

Til denne opgave skal du oprette et nyt projekt i din kodemappe. Det skal være et *WPF App(.NETFramework)* og navngives *WpfAppIntroduktion.* Projektet skal gemmes i din kodemappe, i en mappe der skal navngives *WPF*, her skal du fremover gemme alle dine WPF-projekter.

## 1.1 Opgave 1

Du skal tilrette *MainWindow*, så det får størelsen: Højde-700 og Brede-350 og får titlen *Opgave 1 WPF – Introduktion*.  
Du skal oprette to kolonner. Kolonne nr. 0 skal have en fast brede på 120 og kolonne nr. 1 skal være flydende.   
Du skal nu oprette 10 rækker. Række nr. 0 til nr. 8 skal have en fast højde på 50 og række nr. 9 skal være flydende.  
Du skal nu indsætte en *Button* i kolonne nr. 0 og række nr. 0 til nr. 8. Hver *Button* skal navngives som *buttonOpg12*, *buttonOpg13* … osv. Der må IKKE angives højde og brede på knappen, men i stedet skal der anvendes *Margin* på 10 pix. og både *HorizontalAlignment* & *VerticalAlignment* skal sættes til *Stretch*.  
Knappernes *Content* sættes til *Opgave 1.2*, *Opgave 1.3*, *Opgave 1.4* osv.  
Du skal nu indsætte en ListBox som skal navngives til *listBoxRes*. Den skal placeres i kolonne nr. 1 og række nr. 0. For at sikre at den fylder hele kolonnen, skal der angives en *Grid.RowSpan* med værdien 9. Der må ligeledes IKKE angives fast højde og brede, men der som til knapperne angives en margin, men nu på 5 pix. (sammenlign med hvad gjorde med knapperne og gør det samme her).

Du skal nu tilføje en ny *Class* til projektet, det gøres på samme måde som du gjorde da du arbejdede med *ConsoleApp*. Denne *Class* skal have navnet *ClassBIZ* og skal placeres i en mappe med navnet *BIZ*. I *ClassBIZ* skal du oprette en *constructor*.

I den *Class* der knytter sig til selve *MainWindow* (MainWindow.xaml.cs), skal du lave en instans af din *ClassBIZ* så du kan tilgå de metoder, du snart skal kode for at kunne løse de følgende opgaver. Det gøres på følgende måde: I MainWindow.xaml.cs skriver du, mellem *public partial class MainWindow : Window* og *public MainWindow()* følgende: ClassBIZ classBIZ = new ClassBIZ();

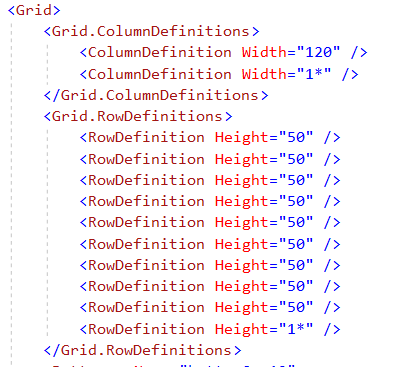
Når du skal tilgå dine metoder i *ClassBIZ*, skal du benytte instansnavnet *classBIZ*.

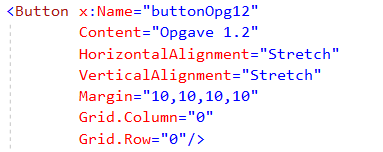
|  |  |
| --- | --- |
| Du skal nu have en GUI der sådan ud: | Din Solution har en struktur som vist her: |
|  |  |

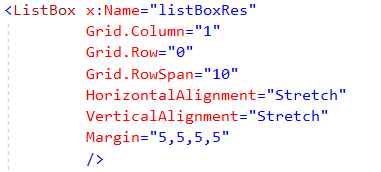
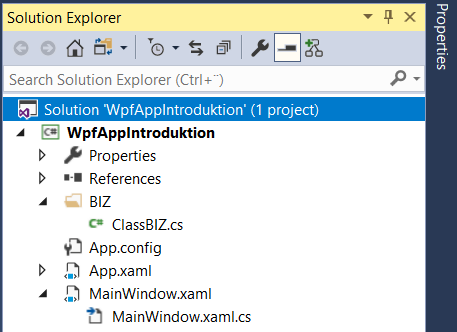
### Hjælp

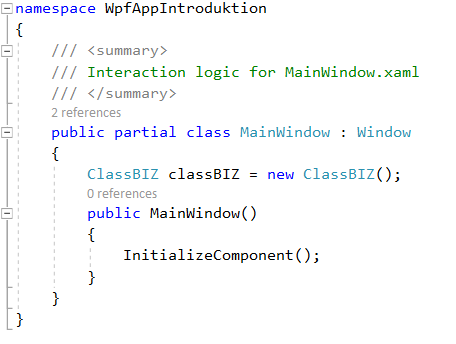
MainWindow:



Grid:  


Button:  


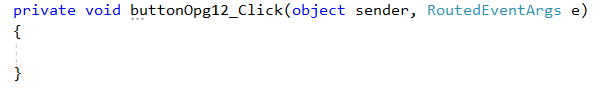
ListBox:  
  
ClassBIZ  


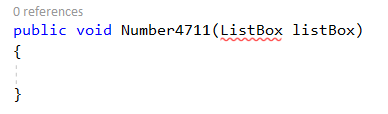
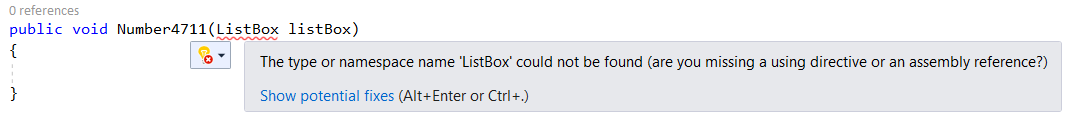
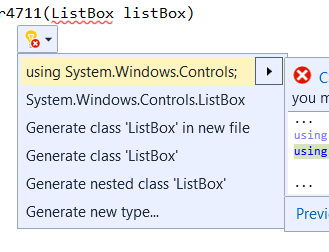
Instans af ClassBIZ:  


## 1.2 Opgave 2

Du skal kode en funktion der udskriver tallene fra 4711 til 4736 i *listBoxRes*. Funktionen skal udføres hver gang du klikker på knappen *buttonOpg12.*   
Derfor skal du nu oprette en *EventHandler* til *buttonOpg12*, der knytter sig til eventet *Click.* Det gøres på følgende måde:

* Marker knappen *buttonOpg12*
* Åben Properties vinduet
* Klik på knappen med lynet i øverste højre hjørne
* Dobb. klik i tekstfeltet ud for navnet Click

Du får nu vist den oprettede EventHandler, som er placeret i MainWindow.xaml.cs, og kan se at EventHandleren er blevet navngivet med kombination af knappens navn og typen af event den skal håndtere.  


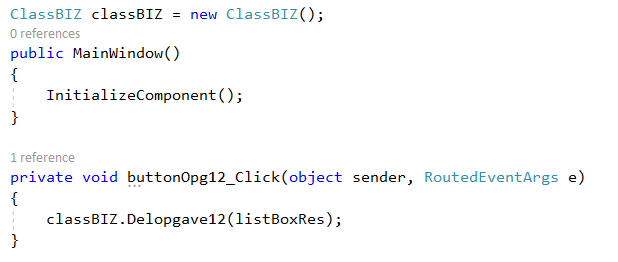
Nu åbner du din ClassBIZ og opretter en ny metode som vist her:  
  
Som du kan se på ovenstående, vil det udløse en fejl, da ClassBIZ ikke ved hvad en ListBox er. Dette løses ved, at du holder cursoren hen over ordet ListBox, så vil følgende blive vist på skærmen:  
  
Nu klikker du pilen i boksen med elpæren og vælger den øverste mulighed i listen over løsningsforslag:  
  
Nu ved din *ClassBIZ* hvad en *ListBox* er, fordi der nu øverst i *ClassBIZ* står:   
using System.Windows.Controls;

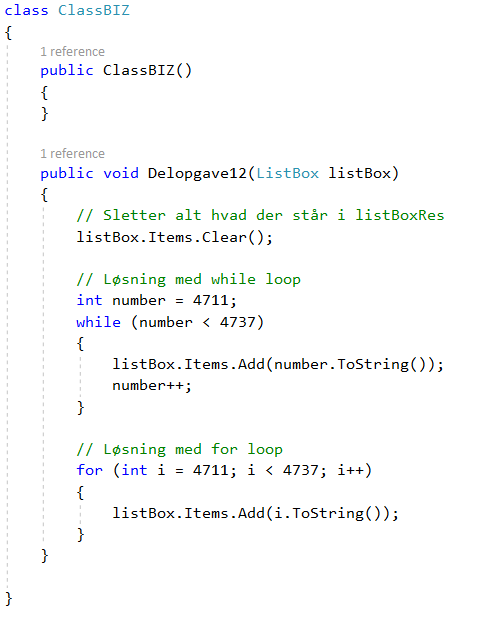
Du skal nu lave noget kode, der gør det muligt at tilføje tallene fra 4711 til 4736 til den ListBox du har placeret på brugergrænsefladen, det kan løses ved at anvende iteration (loop).  
Det betyder at du med fordel kan anvende enten for-loop eller while-loop.  
Når du skal tilføje noget til en ListBox, gøres det på følgende måde:  
listBox.Items.Add(i.ToString());  
Når du har skrevet din kode der indsætter talrækken i listBoxRes, skal du lave et kald fra EventHandleren til din metode i ClassBIZ. Det gøres på følgende måde:  
I EventHandleren der knytter sig til knappen for denne delopgave, skal du skrive følgende:  
classBIZ.Delopgave12(listBoxRes);

* *classBIZ* den instans der er lavet af *ClassBIZ*
* *Delopgave12* er navnet på den metode der skal benyttes i *classBIZ*
* I parentesen angives den parameter der skal sendes med til metoden
  + *listBoxRes* er navnet på den *ListBox* du har placeret på *MainWindow*. Når man sender en parameter med til en metode, er det altid som en reference, altså ikke en ny kopi men en afspejling af det oprindelige objekt. Det har den fordel, at når du manipulerer med den modtagne parameter (spejlingen), vil det straks påvirke det oprindelige objektet der ligger til grund for ”spejlingen”, i det her tilfælde *ListBoxen* med navnet *listBoxRes*.

Test om programmet virker efter hensigten.

### Hjælp

MainWindow.xaml.cs :  


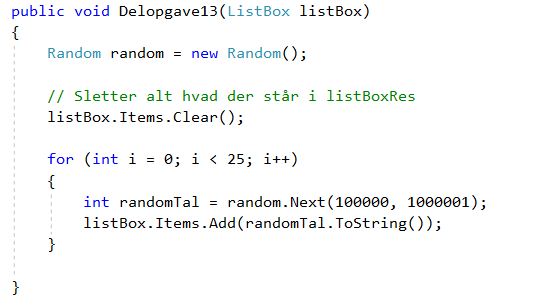
ClassBIZ :  


## Opgave 3

I denne opgave skal du benytte meget af det du lavede i opgave 1.2.

Du skal i *ClassBIZ* lave en metode det opretter 25 tilfældige tal mellem 100.000 og 1.000.000.  
Tallene skal udskrives i *listBoxRes*.  
Metoden skal knyttes til knappen *buttonOpg13*.

### Hjælp



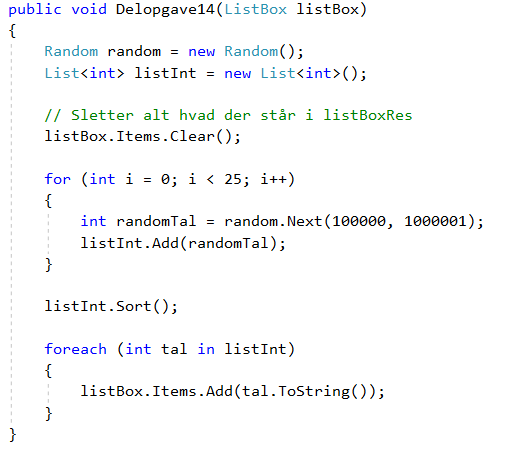
## Opgave 4

I denne opgave skal du som i opgave 1.3 oprette 25 tilfældige tal, men inden de udskrives i *listBoxRes* skal værdierne sorteres i stigende orden.   
opgaven skal knyttes til et klik på *buttonOpg14*.  
Der findes forskellige metoder til at sortere en *collection* af data, man kan skrive sin egen metode til det og her skrive en *BobleSort* funktion. Det er ikke videre indviklet at lave en *BobleSort*, men den er ikke særlig effektiv hvis det drejer sig om en større mængde der skal sorteres. Men hvorfor opfinde den dybe tallerken to gange, hvorfor ikke bare udnytte den super optimerede kode C# stiller til rådighed for os.  
Hvis man lager sin samling af data i en *Collection* af typen *List<T>*, får man adgang til en metode med navnet *Sort()*. Hvis du benytter den, skal du ikke selv oprette en sorteringsmetode til forskellige datatyper, da den kan sortere alle simple datatyper som default er implementeret i C# (string, int, double m.fl.).

Derfor skal du i metoden oprette en instans af datatypen *List<int>* hvor du skal lagre de *random* genereret tal og efterfølgende udføre en *.Sort()* på *List<int> instansen*.

Efter du har gennemført sorteringen, skal data overføreres til *listBoxRes*.

### Tip!

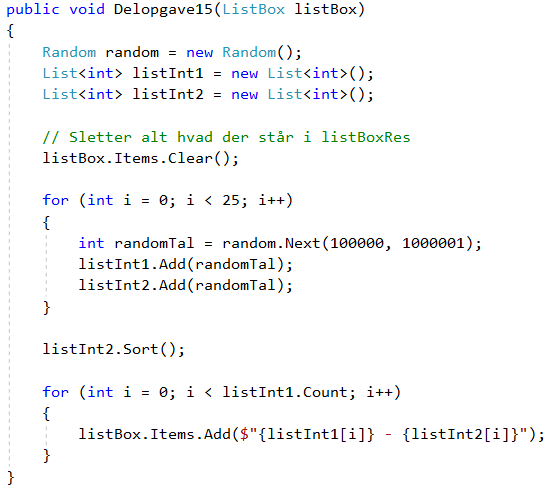


## Opgave 5

I denne opgave skal lave en løsning som er en blanding af opgave 1.3 og 1.4.

Her skal din løsning udskrive listen usorteret og sorteret, så hver linje i *listBoxRes* viser tallets placering i en *List<T>* før og efter en sortering.

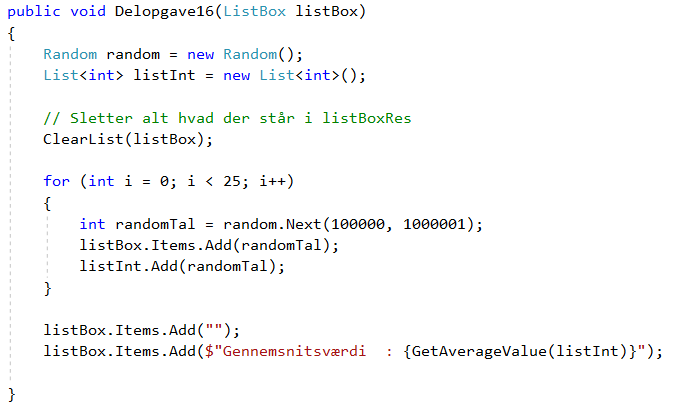
### Tip!

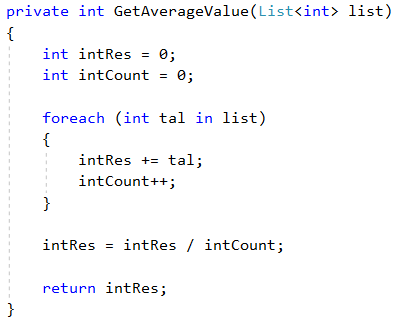


## Opgave 6

Du skal, som i opgave 1.3, i *ClassBIZ* lave en metode det opretter 25 tilfældige tal mellem 100.000 og 1.000.000.  
Tallene skal udskrives i *listBoxRes* efterfulgt af et tomt item og et item med følgende indhold:   
Gennemsnitsværdi : xxxxxxxxx  
Hvor xxxxxxxxx er den beregnede gennemsnitsværdi.  
Til at finde gennemsnitsværdien, skal du oprette en *private* metode der kan returnere datatypen *int.*  
Metoden skal knyttes til knappen *buttonOpg16*.  
Resultatet skal se ud som vist her:  


### Tips!





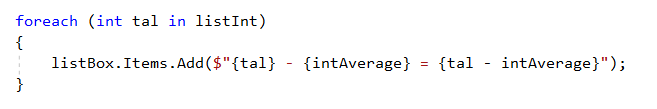
## Opgave 7

Du skal, som i opgave 1.3, i *ClassBIZ* lave en metode det opretter 25 tilfældige tal mellem 100.000 og 1.000.000.  
Inden tallene skal udskrives, skal de sorteres i stigende orden.  
Der skal beregnes en gennemsnitsværdi for samlingen af tal.

Når tallene skal udskrives, skal der skrives tallet – gennemsnitsværdien = beregnet værdi.

Skal se ud som vist her:  


### Tips!



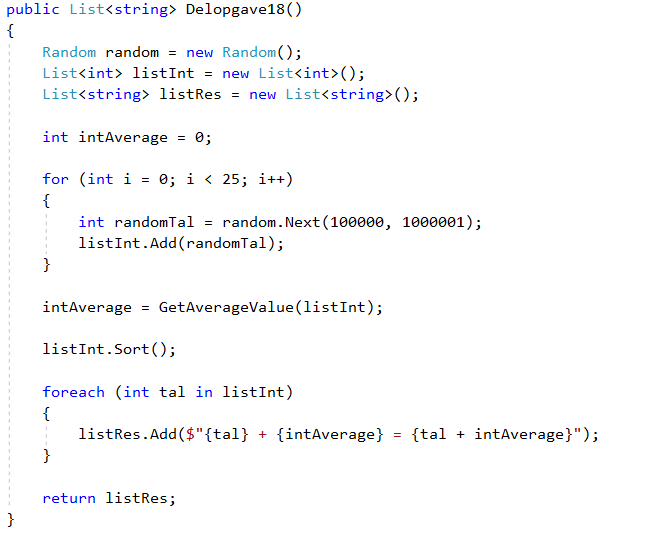
## Opgave 8

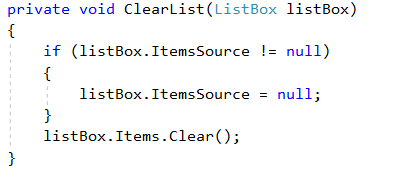
Samme opgave som 1.7, men nu skal du lave en løsning hvor du IKKE sender *listBoxRes* med til metoden i *ClassBIZZ*, men hvor metoden i *ClassBIZZ* skal returnere datatypen *List<string>* som skal knyttes til *listBoxRes* via angivelse af *ItemsSource.*

I eventhandleren der er knyttet til *buttonOpg18* skal du foretage knytning mellem den List<string> der returneres fra metoden og objektet listBoxRes.

Når man benytter *ItemsSource*, kan man ikke længere direkte cleare indholdet af *listBoxRes*, da *listBoxRes* nu er knyttet et objekt via *ItemsSource*. Derfor skal du rette din kode alle de steder hvor benytter *listBoxRes.Clear()* til et metodekald som vil undersøge om *listBoxRes* er knyttet til en *ItemsSource*, hvis den er knyttet, fjernes tilknytningen og indholdet af *listBoxRes* slettes. Metoden du skal oprette skal du have navnet *ClearList* og skal være *private* og modtage datatypen *ListBox*:   
private void ClearList(ListBox listBox)

### Tips!



Metoden ClearList:  


## Opgave 9

Du skal, som i opgave 1.3, i *ClassBIZ* lave en metode det opretter 25 tilfældige tal mellem 100.000 og 1.000.000.  
Inden tallene skal udskrives, skal de sorteres i stigende orden.  
Der skal beregnes en gennemsnitsværdi for samlingen af tal.

Når tallene skal udskrives, skal der skrives: tallet – gennemsnitsværdien = beregnet værdi.

Når resultatet skal indsættes i listBoxRes, skal baggrundsfarven for hver linje afgøres af, om det tilfældige tal er et lige tal eller om det er et ulige tal.

* Lige tal skal give baggrundsfarven – *HotPink*
* Ulige tal skal give baggrundsfarven – *AliceBlue*

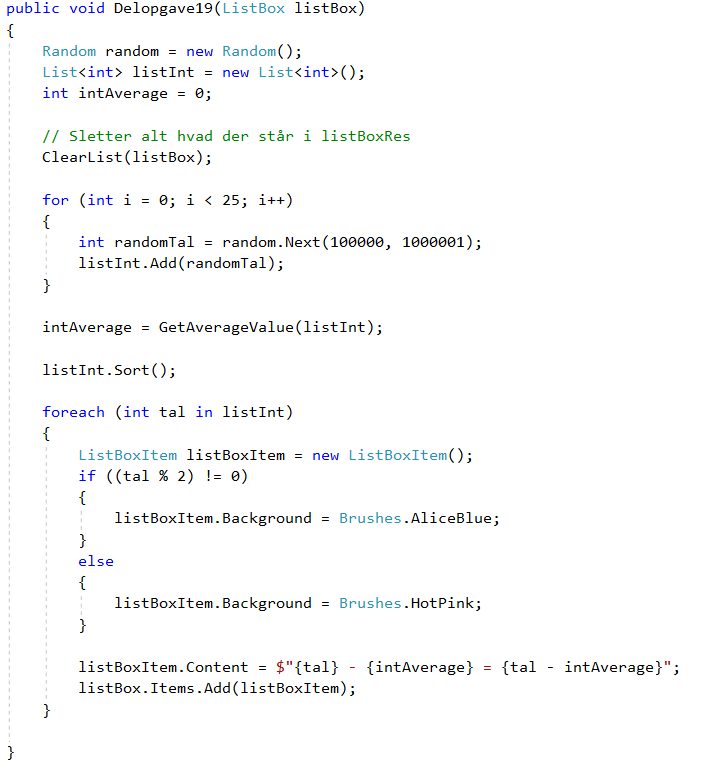
Du skal benytte *modulus* til at beregne om tallet er lige eller ulige.  
For at få mulighed for at sætte baggrundsfarven, skal man benytte et *ListBoxItem*, hvilket du kan få adgang til ved at oprette en ny instans af *ListBoxItem* til hvert af de elementer der skal indsættes i *listBoxRes*.  
ListBoxItem listBoxItem = new ListBoxItem();

Baggrundsfarve sætte på følgende måde:  
listBoxItem.Background = Brushes.AliceBlue;

Teksten tilføjes listBoxItem og listBoxItem tilføjes listBoxRes på følgende måde:  
listBoxItem.Content = $"{tal} - {intAverage} = {tal - intAverage}";

listBox.Items.Add(listBoxItem);

### Tips!



## Opgave 10 – Den svære opgave

Du skal i denne opgave kombinere dine løsninger fra flere af de foregående opgaver.

Du skal, som i opgave 1.3, i *ClassBIZ* lave en metode det opretter 25 tilfældige tal mellem 100.000 og 1.000.000.  
Inden tallene skal udskrives, skal de sorteres i stigende orden.  
Der skal beregnes en gennemsnitsværdi for samlingen af tal.

Når tallene skal udskrives, skal der skrives: tallet – gennemsnitsværdien = beregnet værdi.

Når resultatet skal indsættes i listBoxRes, skal baggrundsfarven for hver linje afgøres af, om det tilfældige tal er et lige tal eller om det er et ulige tal.

* Lige tal skal give baggrundsfarven – *HotPink*
* Ulige tal skal give baggrundsfarven – *AliceBlue*

Du skal benytte *modulus* til at beregne om tallet er lige eller ulige.

Du skal som i opgave 1.8 benytte *ItemsSource* til at knytte data til *listBoxRes*, men knytningen skal foretages i *ClassBIZ*.

Du får ingen hjælp til denne opgave – God arbejdslyst ;O))