(Hint) Agent Assignment 1 Guidance

2022E

Check these slides in lecture 5

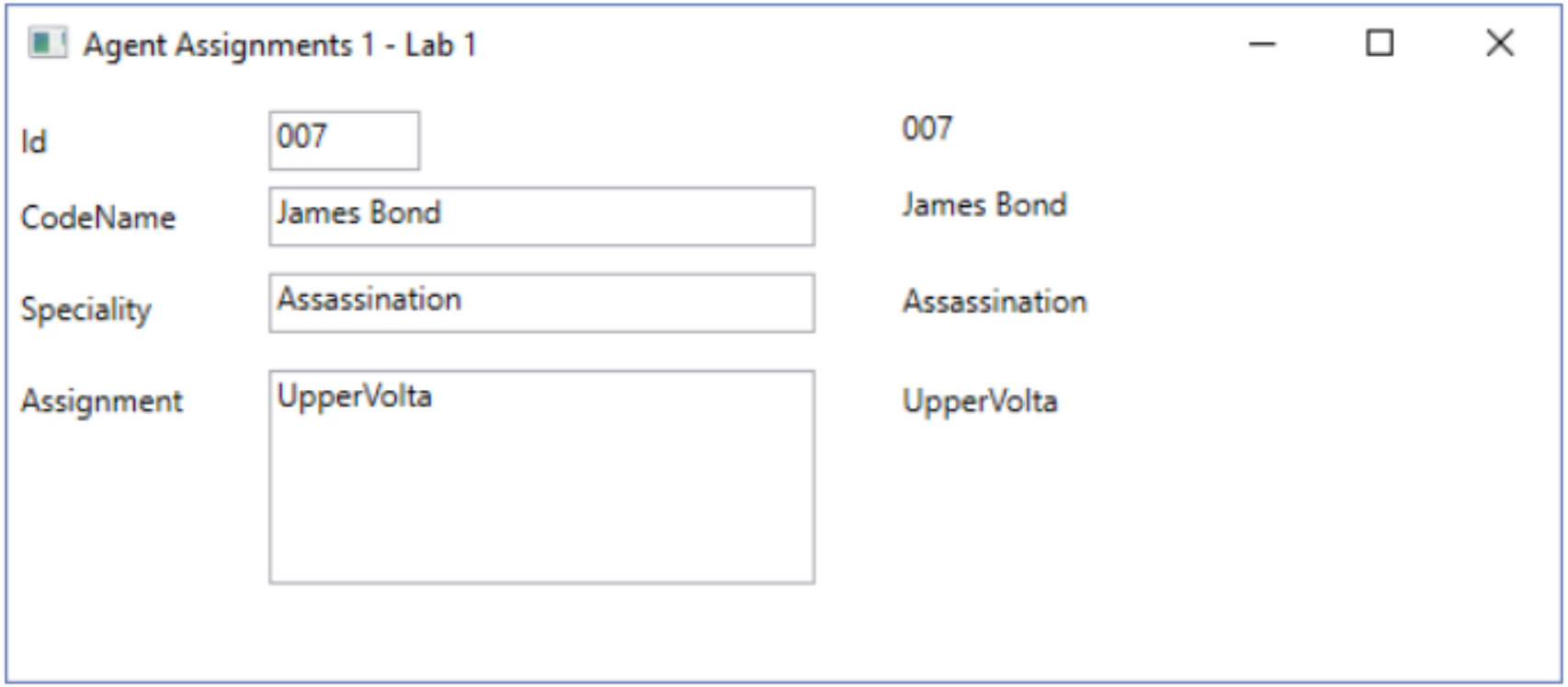
- Slide 14-17 : for understanding theory
- **Slide 18**: **DataContext** Set **in C# code** (slide 15 also about DataContext)
- **Slide 19** : **DataContext** set **in XAML code**
- **Side 21**: Object Changes , **INotifyPropertyChanged** interface
- **Slide 23** – very important code
- Slide 28-31 : **Binding to List Data**
  - Slide 29-30 : observableCollection
  - **Slide 31: IsSynchronizedWithCurrentItem (Demo 11)**

Delopgave1

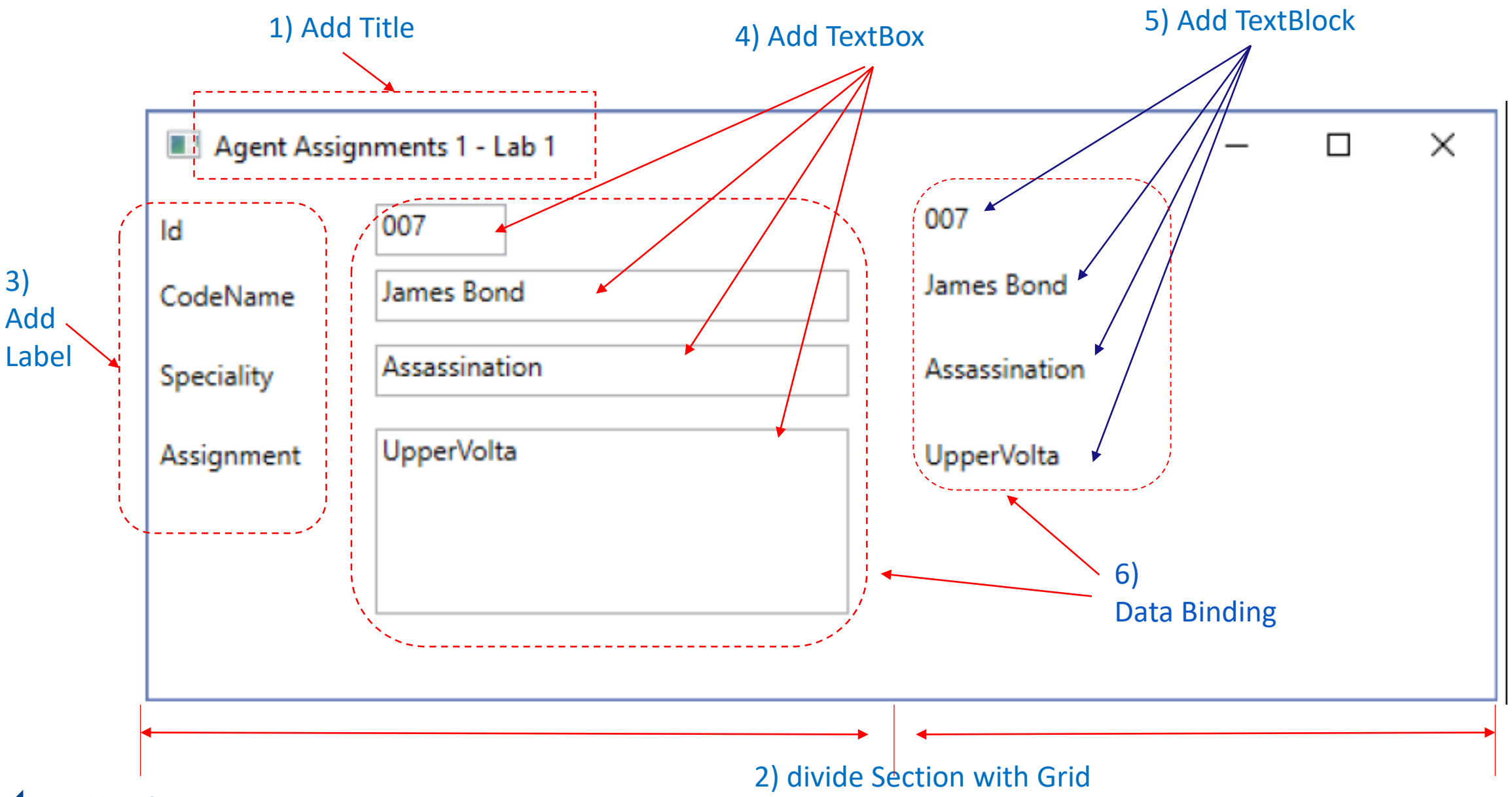
- Lav et WPF program som benytter data binding til at knytte nogle textbokse til attributterne i Agentklassen
- (Agent-klassen er implementeret i filen Agent.cs og kan hentes fra Brightspace).
- For at teste at dataene i objektet virkelig bliver ændret, så tilføjes der også en række tekstblokke som bindes til de samme data

Delopgave 1

Lav et WPF program som benytter data binding til at knytte nogle textbokse til attributterne i Agent-klassen (Agent-klassen er implementeret i filen Agent.cs og kan hentes fra Brightspace). For at teste at dataene i objektet virkelig bliver ændret, så tilføjes der også en række tekstblokke som bindes til de samme data.



In Xaml


1) Add Title
4) Add TextBox
5) Add TextBlock
Agent Assignments 1 - Lab 1
Id
CodeName
Speciality
Assignment
007
James Bond
Assassination
UpperVolta
007
James Bond
Assassination
UpperVolta
3)
Add
Label
6)
Data Binding
2) divide Section with Grid

(ToDo) in MainWindow.xaml.cs

Two things need to remember from **slide 18** :
- Parameter med constructor
- DataContext

1) Add this

```
using AgentAssignment;

public partial class MainWindow : Window
{
    Agent agent = new Agent("007", "James Bond", "Assassination", "UpperVolta");
    0 references
    public MainWindow()
    {
        InitializeComponent();
        DataContext = agent;
    }
}
```

2) parameter med constructor (slide 18)

3) Remember DataContext (Slide18)

An instance of agent class

Use ***DataContext***
(DataContext property is extremely useful if you need to bind several properties of the same object to different elements,)

Agent.cs (**download file** from brightspace)

```
namespace AgentAssignment
{
    4 references
    public class Agent
    {
        string id;
        string codeName;
        string speciality;
        string assignment;

        0 references
        public Agent()
        {
        }

        1 reference
        public Agent(string aId, string aName, string aSpeciality, string aAssignment)
        {
            id = aId;
            codeName = aName;
            speciality = aSpeciality;
            assignment = aAssignment;
        }
```

Data binding in XAML

Data binding from **TextBox** to properties in Agent.cs

Agenet Elements	ToDo: **TextBox** in MainWindow.xaml	Agent.cs (download file)
	(binde deres **Text property til relevant property på Agentklassen**)	**Relevant Properties in Agent.cs file**

ID	`<TextBox Height="` Adjust yourself
	`Margin="`
	`Name="textBox1"`
	`VerticalAlignment="Top"`
	`HorizontalAlignment="Left"`
	`Width="  "`
	`Text="{Binding Path=ID}"/>`
		`public string ID { get { return id; } set { id = value; } }`

CodeName	`<TextBox Height="` Adjust yourself
	`Margin="`
	`Name="textBox2"`
	`VerticalAlignment="Top"`
	`Text="{Binding Path=CodeName}" />`
		`public string CodeName { get { return codeName; } set { codeName = value; } }`

Speciality	`<TextBox Height="` Adjust yourself
	`Margin="`
	`Name="textBox3"`
	`VerticalAlignment="Top"`
	`Text="{Binding Path=Speciality}"/>`
		`public string Speciality { get { return speciality; } set { speciality = value; } }`

Assginment	`<TextBox Height="` Adjust yourself
	`Margin="`
	`Name="textBox4"`
	`AcceptsReturn="True"`
	`TextWrapping="Wrap"`
	`VerticalAlignment="Top"`
	`Text="{Binding Path=Assignment}" />`
		`public string Assignment { get { return assignment; } set { assignment = value; } }`

Data binding "ID"

Data binding "CodeName"

Data binding "Speciality"

Data binding "Assignment"

Data binding from **TextBlock** to properties in Agent.cs

Agenet Elements	ToDo: **TextBlock** in MainWindow.xaml	Agent.cs (download file)
	(binde deres **Text property til relevant property på Agentklassen**)	**Relevant Properties in Agent.cs file**

ID	<**TextBlock** Grid.Column="1"
	Height=
	Width='
	Margin=
	Name="tbId"
	VerticalAlignment="Top" HorizontalAlignment="Left"
	Text="{Binding Path=ID}"/>
		public string ID
		{
		get
		{
		return id;
		}
		set
		{
		id = value;
		}
		}

CodeName	<TextBlock Grid.Column="1"
	Height='
	Width="
	Margin=
	Name="tbCodeName"
	VerticalAlignment="Top" HorizontalAlignment="Left"
	Text="{Binding Path=CodeName}" />
		public string CodeName
		{
		get
		{
		return codeName;
		}
		set
		{
		codeName = value;
		}
		}

Speciality	<TextBlock Grid.Column="1"
	Height="
	Margin='
	Name="tbSpeciality"
	VerticalAlignment "Top"
	Text="{Binding Path=Speciality}" />
		public string Speciality
		{
		get
		{
		return speciality;
		}
		set
		{
		speciality = value;
		}
		}

Assginment	<TextBlock Grid.Column="1"
	Height="21"
	Margin="19,112,0,0"
	Name="tbAssignment"
	VerticalAlignment="Top"
	Text="{Binding Path=Assignment}" />
		public string Assignment
		{
		get
		{
		return assignment;
		}
		set
		{
		assignment = value;
		}
		}

Adjust yourself

Data binding "ID"

Data binding "CodeName"

Data binding "Speciality"

Data binding "Assignment"

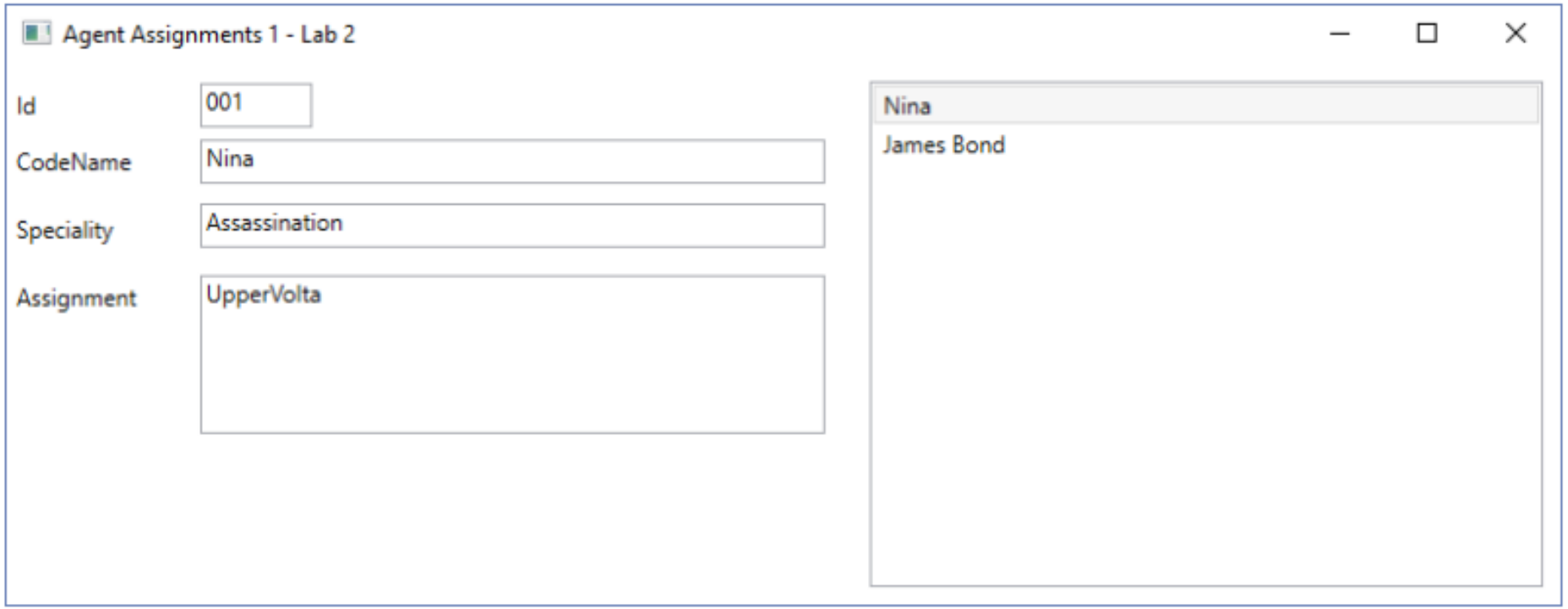
Delopgave 2

Lav et WPF program som benytter data binding til en List data source. I C# koden sættes DataContext til at referere en collection af agenter. Man vælger agent i listboksen. Brug f.eks. DisplayMemberPath til at vise navnet på agenten. Felterne for den valgte agent kan så editeres i tekstboksene til venstre for listboksen. I denne delopgave skal man ikke kunne tilføje nye agenter, kun ændre properties på dem som listen oprettes med.

Delopgave 2

Lav et WPF program som benytter data binding til en List data source. I C# koden sættes DataContext til at referere en collection af agenter. Man vælger agent i listboksen. Brug f.eks. DisplayMemberPath til at vise navnet på agenten. Felterne for den valgte agent kan så editeres i tekstboksene til venstre for listboksen.

I denne delopgave skal man ikke kunne tilføje nye agenter, kun ændre properties på dem som listen oprettes med.



Todo

- Use Grid and **bind with DataContext**

```
agentGrid.DataContext = agents;
```

- How to **bind each control of list from agent (slide 31)**
- How to **synchronize selecteditem in listbox** and show the individual control
- **Bind listbox ItemSource with their property IsSysnchronizeWithCurrentItem** to true. (**Slide 31)**

```
<ListBox ItemsSource="{Binding}"
         IsSynchronizedWithCurrentItem="True"
         DisplayMemberPath="CodeName"
```

In MainWindow.xaml.cs

listen af agenter

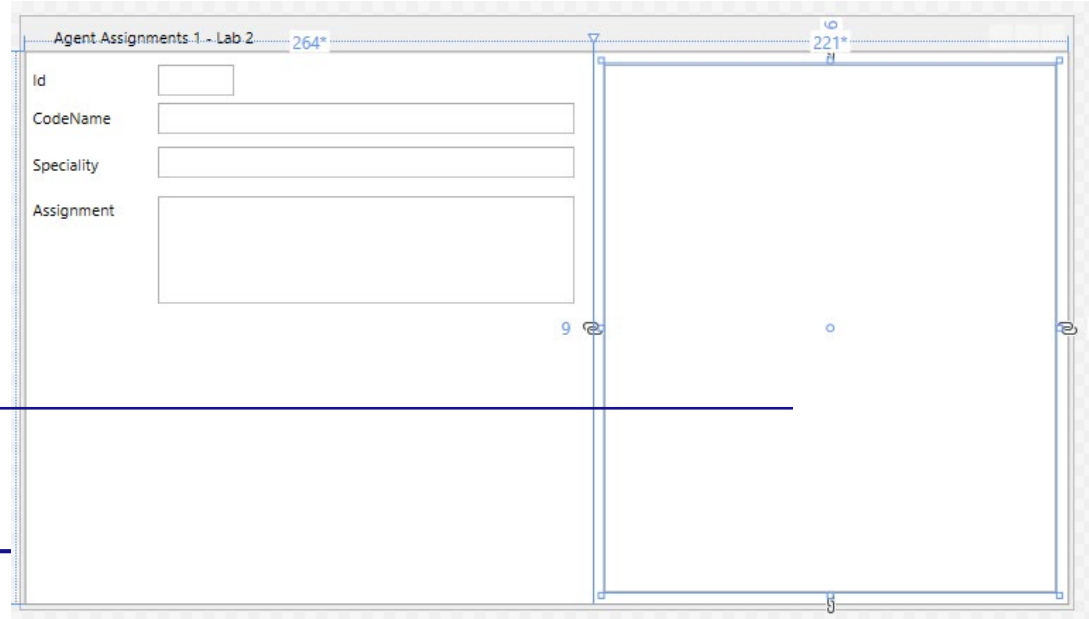
```
public partial class MainWindow : Window
{
    List<Agent> agents = new List<Agent>();
    0 references
    public MainWindow()
    {
        InitializeComponent();
        agents.Add(new Agent() { ID = "001", CodeName = "Roger", Speciality = "Driving", Assignment="Moonraker" });
        agents.Add(new Agent() { ID = "007", CodeName = "Sean", Speciality = "Martiny", Assignment="Goldfinger" });
        agentGrid.DataContext = agents;
    }
}
```

sætte Datacontext på Grid'et i stedet for på vinduet, givet grid'et et Name i XAML-koden

DataContext til listen af agenter, så kan det gøres meget smart ved at binde listboksens **ItemSource** (uden at angive Path, da den skal bindes til selve objektet i listen) samt sættes dens property **IsSynchronizedWithCurrentItem til True**.

In Xaml

```
<ListBox ItemsSource="{Binding}"
         IsSynchronizedWithCurrentItem="True"
         DisplayMemberPath="CodeName"
         Name="lbxAgents"
         Grid.Column="1"
         Margin="
         />
```



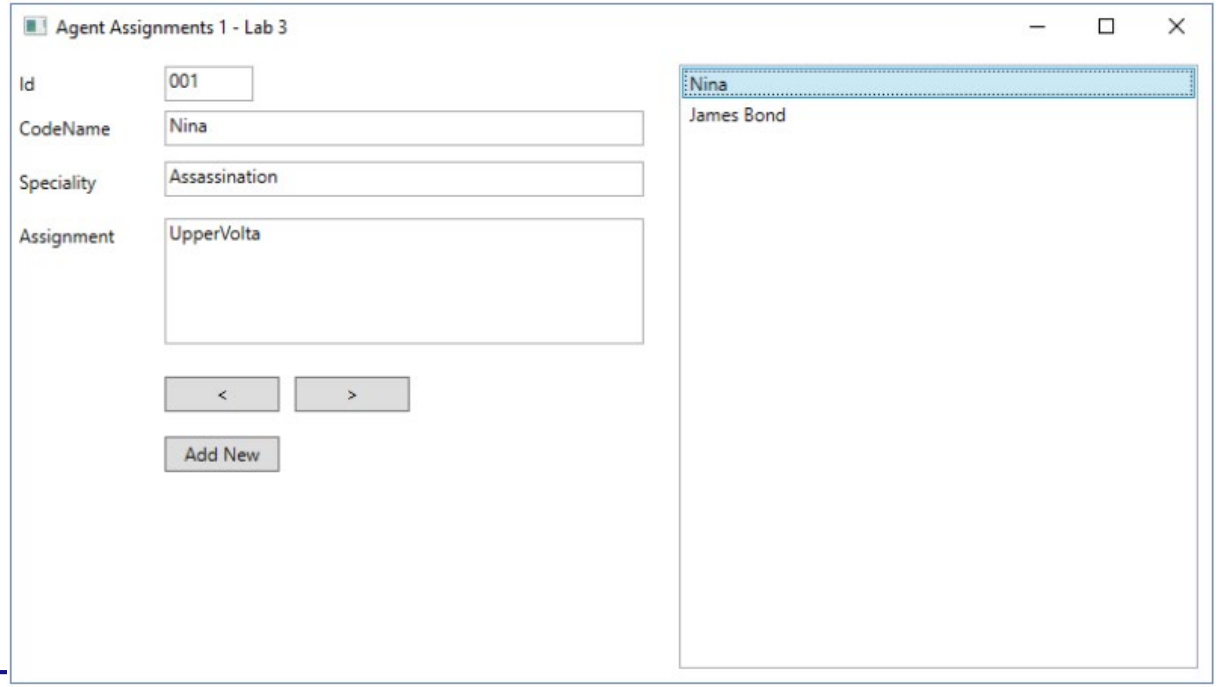
Delopgave3

Delopgave 3

Lav et WPF program som benytter data binding til en List data source, således at man kan editerer den aktuelle agent, og som indeholder knapper til at navigere gennem listen samt giver mulighed for tilføjelse af nye agenter. Sæt datakonteksten i XAML.

HJÆLP:

1. Lav en MainWindowViewModel klasse som MainWindows DataContext initieres med i XAML.
2. MainWindowViewModel skal have en property som returnerer listen af agenter.
   Og denne liste skal være af typen ObservableCollection, så brugergrænsefladen automatisk opdateres når der tilføjes nye agenter.
3. ListBoxens ItemSource skal bindes til denne property.
4. Da DataContext nu peger på en ViewModel, så virker databindingen for tekstboksene ikke i den simple udgave, som var ok til lab 1 og 2. For at få dem til at virke igen, så tilføjes en property på MainWindowViewModel med navnet CurrentAgent og typen Agent, som implementerer get og set - og hvor der i set kaldes NotifyPropertyChanged.
   Til at styrer hvem der er CurrentAgent bruges listboksen, så lisboksens property SelectedItem skal bindes til CurrentAgent.
   De forskellige TextBokse skal så have opdateret deres bindings, så den for eksempel for Id-tekstboksen komme til at hedde `Path="CurrentAgent.ID"`.
5. Man kan f.eks. navigere igennem listen ved at sætte selectedIndex på listBoxen. Dette gøres i en eventhandler for frem og tilbage knapperne.
6. Eventhandleren for "Add New" skal kalde en metode i MainWindowViewModel som tilføjer en ny agent til listen. Og efterfølgende skal denne nye agent automatisk vælges i listboksen, så brugeren kan indtaste data for den nye agent.



Fra kommentarer

I denne opgave er der 3 udfordringer:

1. Man skal sætte DataContext i XAML-koden,
2. Man skal kunne navigerer frem og tilbage i listen ved tryk på en knap
3. Man skal kunne oprette nye agenter.

First Check **kommentarer pdf** in brightspace

Fra komentarer pdf

1: I hjælp til opgaven står der at man skal bruge en ViewModel til løsning af opgaven, så jeg tilføjer klassen MainWindowViewModel, og i den opretter jeg en ObservableCollection af Agents, og en property, som kan returnere denne liste.
I XAML-koden oprettes en instans af MainWindowViewModel i Winduets DataContext:

```
<Window.DataContext>
    <local:MainWindowViewModel />
</Window.DataContext>
```

For at vist listen af agenter i listboxen, så skal dens ItemSource bindes til Agents-propertyen:

```
ItemsSource="{Binding Path=Agents}"
```

For at få textboxene til at virke igen, så skal MainWindowViewModel tilføjes en property CurrentAgent - og det er vigtigt at den notifier.

```
public Agent CurrentAgent
{
    get { return currentAgent; }
    set
    {
        if (currentAgent != value)
        {
            currentAgent = value;
            NotifyPropertyChanged();
        }
    }
}
```

Og så skal listboksens SelectedItem bindes til den, og det samme skal tekstboxene.

Fra komentarer pdf

2: I knappernes eventhandlere implementerer jeg navigeringen ved at ændre SelectedIndex for listboksen.

```
void btnForward_Click(object sender, RoutedEventArgs e)
{
    if (lbxAgents.SelectedIndex < lbxAgents.Items.Count - 1)
        lbxAgents .SelectedIndex = ++lbxAgents.SelectedIndex;
}
```

3: Det store problem med at indsætte en ny agent er, at listen er oprettet i Xaml-koden, og at man ikke kan give den et navn, således at den kan tilgås som et almindeligt datamedlem i koden. Men vi ved at Winduets DataContext indeholder vores MainWindowViewModel objekt så vi kan derigennem få en reference som vi kan typecaste til MainWindowViewModel og på den kan vi kalde en metode, som kan tilføje vores nye agent.

Af hensyn til brugervenlighed flytter jeg selected index til den nye agent og flytter focus til det første indtastningsfelt.

```
private void BtnAddNew_Click(object sender, RoutedEventArgs e)
{
    var vm = DataContext as MainWindowViewModel;
    vm.AddNewAgent();
    lbxAgents.SelectedIndex = lbxAgents.Items.Count - 1;
    tbxId.Focus();
}
```

MainWindow.xaml.cs

```
namespace Lab3
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    2 references
    public partial class MainWindow : Window
    {
        0 references
        public MainWindow()
        {
            InitializeComponent();
        }

        1 reference
        private void BtnBack_Click(object sender, RoutedEventArgs e)
        {
            if (lbxAgents.SelectedIndex > 0)
                lbxAgents.SelectedIndex = --lbxAgents.SelectedIndex;
        }

        1 reference
        private void BtnForward_Click(object sender, RoutedEventArgs e)
        {
            if (lbxAgents.SelectedIndex < lbxAgents.Items.Count - 1)
                lbxAgents.SelectedIndex = ++lbxAgents.SelectedIndex;
        }

        1 reference
        private void BtnAddNew_Click(object sender, RoutedEventArgs e)
        {
            var vm = DataContext as MainWindowViewModel;
            vm.AddNewAgent();
            lbxAgents.SelectedIndex = lbxAgents.Items.Count - 1;
            tbxId.Focus();
        }

    }
}
```

MainWindowViewModel.cs (create a new file)

```
using AgentAssignment;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace Lab3
{
    2 references
    public class MainWindowViewModel : INotifyPropertyChanged
    {
        ObservableCollection<Agent> agents;

        0 references
        public MainWindowViewModel()
        {
            agents = new ObservableCollection<Agent>();
            agents.Add(new Agent("001", "Nina", "Assassination", "UpperVolta"));
            agents.Add(new Agent("007", "James Bond", "Martinis", "North Korea"));
            CurrentAgent = agents[0];
        }

        Properties

        Methods

        INotifyPropertyChanged implementation
    }
}
```

refer lecture Slide 29-30 (ObservableCollection)

1)

2)

3)

1) tilføjer klassen MainWindowViewModel, og
2) i den opretter en ObservableCollection af Agents, og
3) en property, som kan returnere denne liste.

next slides

Tilføje en Property **CurrentAgent** in MainWindowViewModel

For at vist listen af agenter i listboxen,
så skal **MainWindowViewModel** tilføjes en property CurrentAgent - og det er vigtigt at den notifier.

```
#region Properties

Agent currentAgent = null;

1 reference
public Agent CurrentAgent
{
    get { return currentAgent; }
    set
    {
        if (currentAgent != value)
        {
            currentAgent = value;
            NotifyPropertyChanged();
        }
    }
}

0 references
public ObservableCollection<Agent> Agents
{
    get
    {
        return agents;
    }
}

#endregion
```

```
using AgentAssignment;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace Lab3
{
    2 references
    public class MainWindowViewModel : INotifyPropertyChanged
    {
        ObservableCollection<Agent> agents;

        0 references
        public MainWindowViewModel()
        {
            agents = new ObservableCollection<Agent>();
            agents.Add(new Agent("001", "Nina", "Assassination", "UpperVolta"));
            agents.Add(new Agent("007", "James Bond", "Martinis", "North Korea"));
            CurrentAgent = agents[0];
        }

        Properties

        Methods

        INotifyPropertyChanged implementation
    }
}
```

Refer lecture slide 21, slide 23 (INotifyPropertyChanged)

```
#region Methods

1 reference
public void AddNewAgent()
{
    agents.Add(new Agent());
}

#endregion
```

Man skal kunne oprette nye agenter.

In XAML

```
<TextBox Height="23"
         Margin="100,10,0,0"
         Name="tbxId"
         VerticalAlignment="Top"
         HorizontalAlignment="Left"
         Width="58"
         Text="{Binding Path=CurrentAgent.ID}"
         />
<TextBox Height="23" Margin="100,39,14,0" Name="textBox2"
         VerticalAlignment="Top"
         Text="{Binding Path=CurrentAgent.CodeName}" />
<TextBox Height="23" Margin="100,72,14,0" Name="textBox3"
         VerticalAlignment="Top"
         Text="{Binding Path=CurrentAgent.Speciality}"/>
<TextBox Margin="100,109,14,0"
         Name="textBox4"
         Text="{Binding Path=CurrentAgent.Assignment}"
         Height="82"
         AcceptsReturn="True"
         TextWrapping="Wrap"
         VerticalAlignment="Top"
         />
```

1) 2) 3) 4)

```
<Button Height="23" Margin="100,212,0,0"
        VerticalAlignment="Top" HorizontalAlignment="Left"
        Name="btnBack" Content="&lt;" Width="75"
        Click="BtnBack_Click"
        />
<Button Height="23" Margin="185,212,0,0"
        VerticalAlignment="Top" HorizontalAlignment="Left"
        Name="btnForward"  Width="75"
        Content="&gt;"
        Click="BtnForward_Click"
        />
<Button Height="23" Margin="100,251,0,0"
        VerticalAlignment="Top" HorizontalAlignment="Left"
        Width="75"
        Name="btnAddNew" Content="Add New"
        Click="BtnAddNew_Click"
        />
```

In MainWidnowViewModel.cs

```
public class MainWindowViewModel : INotifyPropertyChanged
{
    ObservableCollection<Agent> agents;

    0 references
    public MainWindowViewModel()
    {
        agents = new ObservableCollection<Agent>();
        agents.Add(new Agent("001", "Nina", "Assassination", "UpperVolta"));
        agents.Add(new Agent("007", "James Bond", "Martinis", "North Korea"));
        CurrentAgent = agents[0];
    }
```

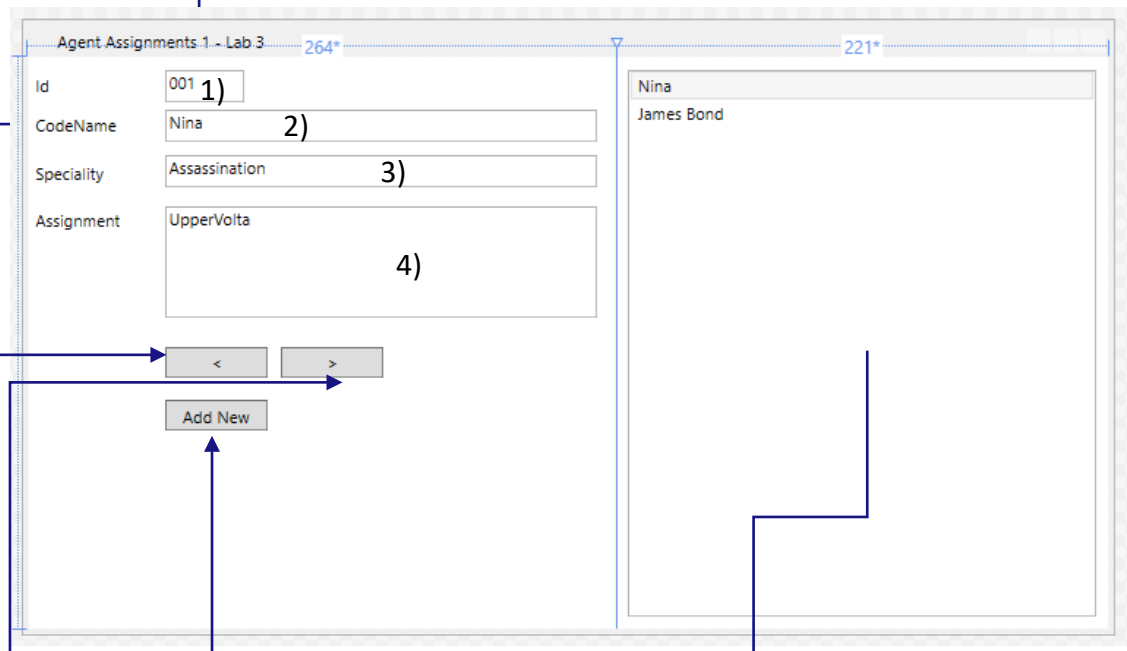
```
#region Properties

Agent currentAgent = null;

1 reference
public Agent CurrentAgent
{
    get { return currentAgent; }
    set
    {
        if (currentAgent != value)
        {
            currentAgent = value;
            NotifyPropertyChanged();
        }
    }
}

0 references
public ObservableCollection<Agent> Agents
{
    get
    {
        return agents;
    }
}

#endregion
```




```
<ListBox ItemsSource="{Binding Path=Agents}"
         SelectedItem="{Binding Path=CurrentAgent}"
         IsSynchronizedWithCurrentItem="True"
         DisplayMemberPath="CodeName"
         Name="lbxAgents"
         Grid.Column="1"
         Margin="9"
         />
```

For at vist listen af agenter i listboxen, så skal dens ItemSource bindes til Agents-propertyen

Og så skal listboksens SelectedItem bindes til den, og det samme skal tekstboxene.