IUT ANNECY UNIVERSITÉ SAVOIE MONT BLANC

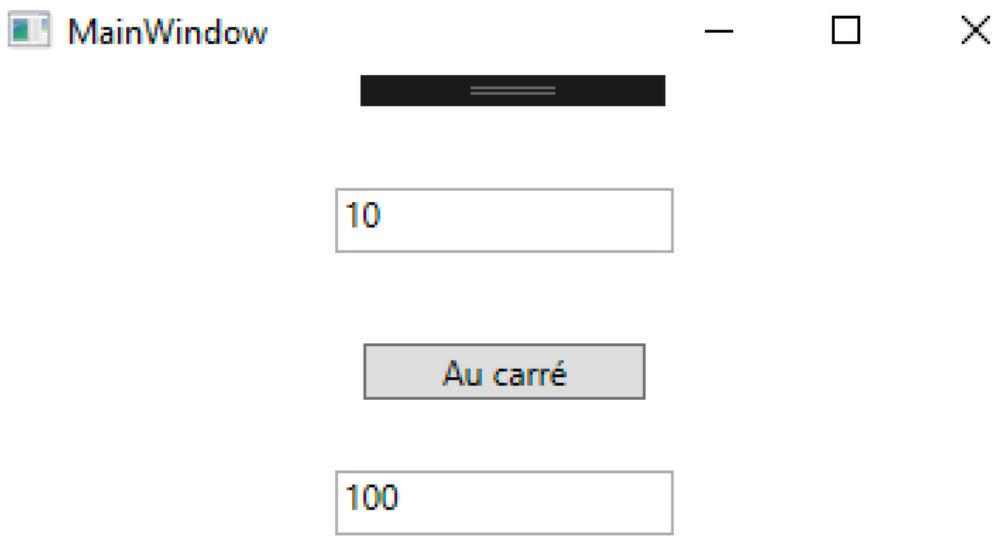
INFO

RAPPEL : BINDING C#

Vincent COUTURIER

Module R3.04

Exemple




```
<Grid Margin="0,0,2,0">
        <TextBox … Text="{Binding Nb1}"/>
         <Button x:Name="ButtonCarre" Margin="0,95,4,0"
Content="Au carré" VerticalAlignment="Top" Width="100"
Click="ButtonCarre_Click"/>
        <TextBox … Text="{Binding Resultat}"/>
</Grid>
```

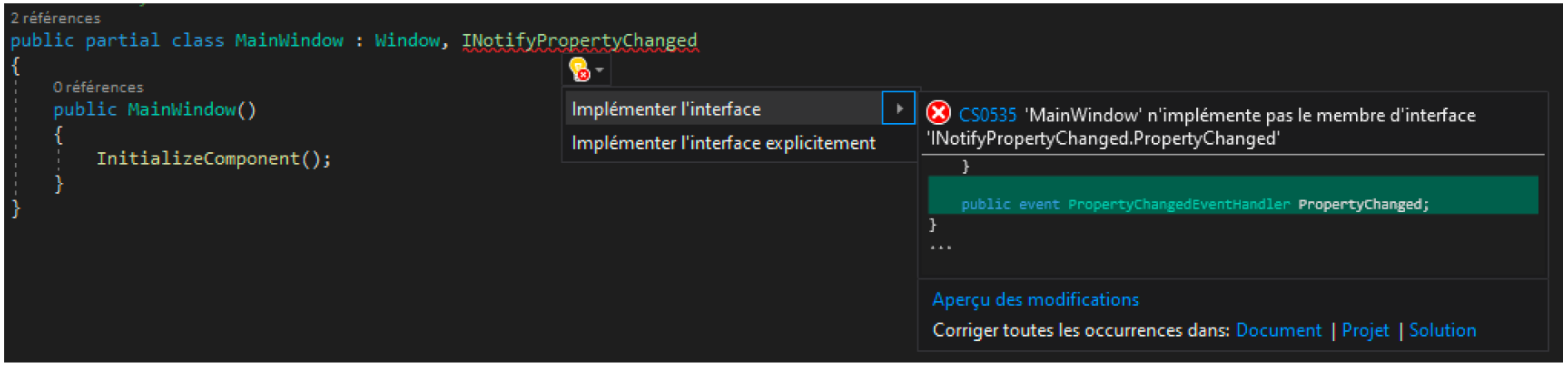
Seul le bouton est nommé (`x:Name`). Les TextBox ne sont pas nommés car binding de données.

Code behind

```
public partial class MainWindow : Window, INotifyPropertyChanged {
        public event PropertyChangedEventHandler PropertyChanged;
        public MainWindow()
        {
            InitializeComponent();
            this.DataContext = this;
        }
        public double Nb1 { get; set; } //Property Nb1
        private Double resultat;
        public Double Resultat { //Property Resultat
            get { return resultat; }
            set {
                resultat = value;
                OnPropertyChanged("Resultat"); } }
        protected void OnPropertyChanged(string name)
        {
            PropertyChangedEventHandler handler = PropertyChanged;
            if (handler != null) {
                handler(this, new PropertyChangedEventArgs(name)); }
        }
        …
    }
```

Code behind

- La classe :
  - Contient les properties qui sont bindées dans la vue XAML.
  - Doit définir le contexte du binding à la fenêtre XAML courante afin de récupérer les valeurs saisies dans la vue : `this.DataContext = this;` Il est courant de faire cette affectation dans le constructeur (**après** `InitializeComponent();`)
  - Doit, afin de mettre à jour la vue :
    - Implémenter l'interface `INotifyPropertyChanged`
    - Créer l'évènement `PropertyChanged` de type `PropertyChangedEventHandler`. Il est possible de le générer de la façon suivante :



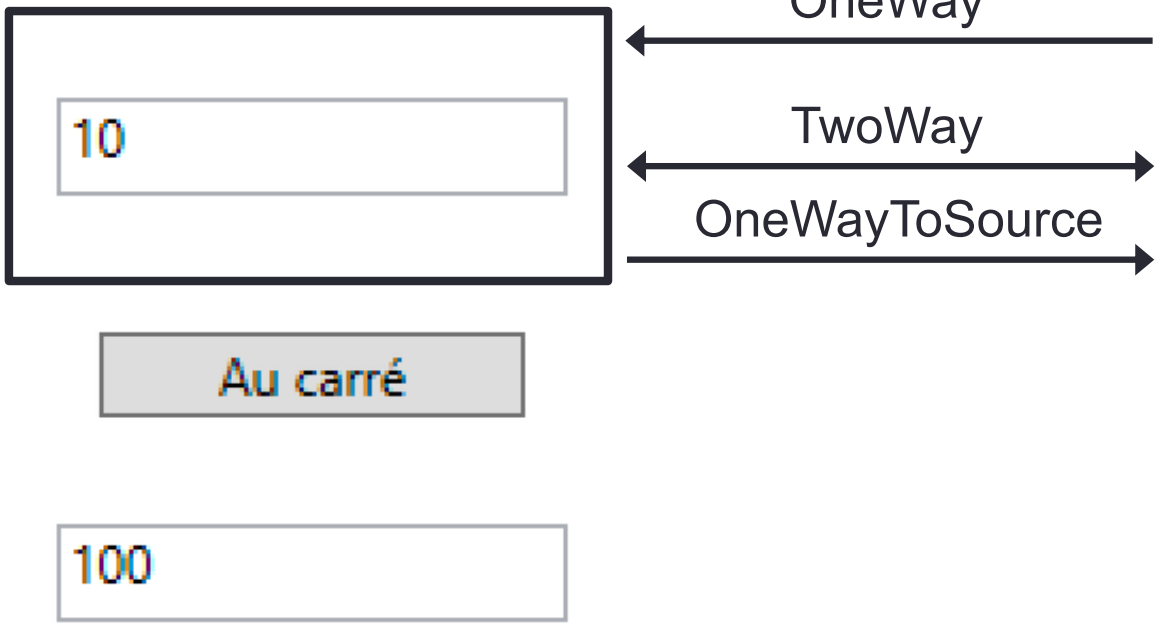
  - Définir la méthode `OnPropertyChanged(string name)` qui prend en paramètre le nom de la property à mettre à jour dans la vue XAML (notification). Pour chaque property pour laquelle vous souhaitez notifier les changements, vous devez ensuite appeler `OnPropertyChanged` chaque fois que la propriété est mise à jour.

Rappels : binding

- Le binding se fait sur un attribut XAML d'un contrôle, par exemple `Text`.
- Le binding `<TextBox … Text="{Binding Nb1}"/>` peut aussi s'écrire :

  `<TextBox … Text="{Binding Path=Nb1, Mode=TwoWay}"/>`

  Car le mode TwoWay est le mode par défaut :




```
public int Nb1 { get; set; }
```

```
//TwoWay ou OneWayToSource
private int nb1;

public int Nb1
{
    get { return nb1;}
    set {
        nb1 = value;
        OnPropertyChanged("Nb1");
        // OU
        //OnPropertyChanged(nameof(Nb1));
    }
}
```

Rappels : binding

- Une `List` n'est pas bindable, il faut créer une `ObservableCollection`.
- Pour alimenter l'ObservableCollection à partir d'une List, cela se fait dans le constructeur d'ObservableCollection :

```
private ObservableCollection<int> collecEntiers;

public ObservableCollection<int> CollecEntiers
{
    get { return collecEntiers; }
    set {
        collecEntiers = value;
        OnPropertyChanged(nameof(CollecEntiers));
    }
}

public Class1()
{
    CollecEntiers= new ObservableCollection<int>(new List<int>() { 1, 2, 5, 7, 8, 10 });
}
```