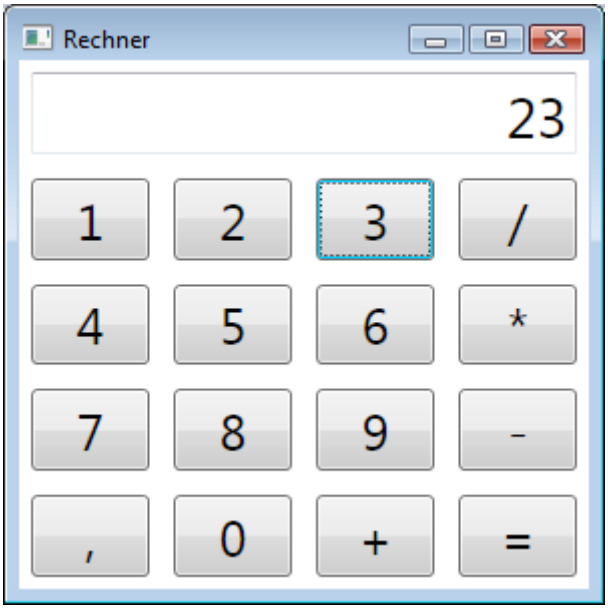
Windowsprogrammierung – WiSe17

Übung 5

Aufgabe 5.1 Taschenrechner

Schreiben Sie ein WPF-Programm für einen einfachen Taschenrechner (+-*/), siehe untenstehendes Bild. Die Eingabe soll nur über Mausklicks erfolgen.



Aufgabe 5.2 XAML und C#, Teil 1

Geben Sie den C#-Code an, der zu dem folgenden XAML-Code äquivalent ist:

```
<ListBox Width="500" Height="500">
    <ListBoxItem>Hallo</ListBoxItem>
    <ListBoxItem Content="Tschüss" />
    <ListBoxItem>
        <ListBoxItem.Content>
```

```
                <Button Content="Klick mich"/>
            </ListBoxItem.Content>
        </ListBoxItem>
        <ListBoxItem>
            <CheckBox IsChecked="True">
                <RadioButton>Ein Radio-Button!</RadioButton>
            </CheckBox>
        </ListBoxItem>
        <Button Content="Es muss nicht immer ein ListBoxItem sein"/>
    </ListBox>
```

Beispielsweise ist für den XAML-Code

```
<Button Width="200" Height="50">
    Hallo Button
</Button>
```

dieser C#-Code äquivalent:

```
Button b = new Button();
b.Width = 200;
b.Height = 50;
b.Content = "Hallo Button";
```

Aufgabe 5.3 XAML und C#, Teil 2

Schreiben Sie ein WPF-Programm, das die folgenden Definitionen enthält:

- Klasse **Adresse** mit den folgenden Membern:
  - Eigenschaften (jeweils öffentlicher Lese- und Schreibzugriff) **Straße** (*string*), **Hausnummer** (*ushort*), **Ort** (*string*), **Postleitzahl** (*ulong*) sowie zugehörige Felder
  - Öffentlicher Standardkonstruktor
  - Überschriebene *ToString*-Methode
- Enumeration **Geschlecht** mit den Enumeratoren **Männlich**, **Weiblich**, **Unbekannt**
- Klasse **Person** mit den folgenden Membern:
  - Eigenschaften (jeweils öffentlicher Lese- und Schreibzugriff) **VollerName** (*string*), **Geschlecht** (**Geschlecht**), **Adresse** (**Adresse**) sowie zugehörige Felder
  - Öffentlicher Standardkonstruktor
  - Überschriebene *ToString*-Methode

Definieren Sie diese Elemente innerhalb des Namensraums **MeineKlassen**.
Die graphische Oberfläche soll aus einem *StackPanel* bestehen, welches wiederum 5 *Label*s beinhaltet. Weisen Sie im XAML-Code der Inhaltseigenschaft der *Label*s mittels der Content-Property-Syntax folgendes zu:

- **Label 1**: Eine Instanz der **Adresse**-Klasse mit sinnvollen Eigenschaftswerten.
- **Label 2**: Eine Instanz der **Person**-Klasse mit sinnvollen Eigenschaftswerten. Weisen Sie der **Adresse**-Eigenschaft eine **Adresse** mit Hilfe der Property-Element-Syntax zu.
- **Label 3**: Legen Sie zunächst die Eigenschaft **VollerName** der **Person**-Klasse als Inhaltseigenschaft dieser Klasse fest. Der Inhalt von **Label 3** soll ebenfalls eine Instanz der *Person*-Klasse sein. Verwenden Sie in diesem Fall die Content-Property-Syntax, um den Wert der Eigenschaft **VollerName** zu setzen.
- **Label 4**: Im Anhang finden Sie eine *TypeConverter*-Implementierung für die Klasse **Adresse**, mit deren Hilfe eine Zeichenkette in dem Format

  "**Von-Ossietzky-Straße 99, 37085 Göttingen**"

  in ein entsprechendes **Adresse**-Objekt verwandelt werden kann. Binden Sie diesen Code in ihr Projekt ein. Zeichnen Sie die Klasse **Adresse** mit dem *TypeConverter*-Attribut aus:

```
[System.ComponentModel.TypeConverter(typeof(AdressConverter))]
class Adresse
{ ... }
```

  Der Inhalt von **Label 4** soll ebenfalls eine Instanz der **Person**-Klasse sein. Legen Sie den Wert der **Adresse**-Eigenschaft mit Hilfe der Attribut-Syntax fest, was dank des *TypeConverter*s nun möglich ist.

- **Label 5**: Definieren Sie im Namensraum **MeineKlassen** eine weitere Klasse **Namen**, die ausschließlich aus öffentlichen, statischen Eigenschaften mit Lesezugriff vom Typ *string* besteht. Inhalt und Bezeichnung dieser Eigenschaften sind Ihnen überlassen. Auch **Label 5** soll eine Instanz der **Person**-Klasse zum Inhalt haben. Weisen Sie der Eigenschaft **Name** mit Hilfe der Static-Markup-Erweiterung einen Eigenschaftswert aus der Klasse **Namen** zu.

Anhang zu Aufgabe 5.3: Implementierung eines TypeConverters für die Klasse Adresse

```
class AdressConverter : System.ComponentModel.TypeConverter
    {
        public override bool CanConvertFrom(System.ComponentModel.ITypeDescriptorContext
context, Type sourceType)
        {
            if (sourceType == typeof(string))
                return true;
            return base.CanConvertFrom(context, sourceType);
        }
        public override object ConvertFrom(System.ComponentModel.ITypeDescriptorContext
context, System.Globalization.CultureInfo culture, object value)
        {
            try
            {
                char[] ziffern = "0123456789".ToCharArray();
                string strValue = ((string)value).Trim();
                Adresse adresse = new Adresse();
                int hausnummerPos = strValue.IndexOfAny(ziffern);
                adresse.Straße = strValue.Substring(0, hausnummerPos - 1);
                int kommaPos = strValue.IndexOfAny(new char[]{',',' '}, hausnummerPos);
                adresse.Hausnummer = ushort.Parse(strValue.Substring(hausnummerPos,
kommaPos - hausnummerPos));
                int postleitzahlPos = strValue.IndexOfAny(ziffern, kommaPos);
                int ortPos = strValue.IndexOf(' ', postleitzahlPos) + 1;
                adresse.Postleitzahl = ulong.Parse(strValue.Substring(postleitzahlPos,
ortPos - postleitzahlPos - 1));
                adresse.Ort = strValue.Substring(ortPos, strValue.Length - ortPos);
                return adresse;
            }
            catch
            {
                throw new Exception(string.Format("Der Wert {0} kann nicht in den Typ {1}
umgewandelt werden.", value, typeof(Adresse)));
            }
        }

    }
```

HAWK Fakultät Naturwissenschaften und Technik Göttingen

Aufgabe 5.4

Schreiben Sie ein WPF-Programm, dessen Benutzeroberfläche ein *StackPanel* enthält. Dieses Element wird in der Vorlesung später ausführlicher behandelt. Alles, was Sie jetzt wissen müssen, ist, dass es sich hierbei um eine Art Container handelt, in dem beliebige graphische Elemente vertikal untereinander oder horizontal nebeneinander angeordnet werden. Platzieren Sie vier *Button*s innerhalb des *StackPanel*s, indem Sie sie im Visual Studio-Designer über das *StackPanel* ziehen. Die 4 *Button*s sollen auf Benutzereingaben wie folgt reagieren:

- Wird der Mauszeiger über **Button 1** bewegt (*MouseEnter*-Event), soll die *Visibilty*-Eigenschaft von **Button 3** auf den Wert *Visibility.Hidden* gesetzt werden. Verlässt der Mauszeiger **Button 1** (*MouseLeave*-Event), soll die *Visiblity* von **Button 3** wieder auf den Wert *Visiblity.Visible* gesetzt werden.
- Wird der Mauszeiger über **Button 2** bewegt, soll die *Visibilty*-Eigenschaft von **Button 3** auf den Wert *Visibility.Collapsed* gesetzt werden. Verlässt der Mauszeiger **Button 1** soll die *Visiblity* von **Button 3** wieder auf den Wert *Visiblity.Visible* gesetzt werden. Beachten Sie den Unterschied zur ersten Variante.
- Klickt der Benutzer auf **Button 3**, wird die *IsEnabled*-Eigenschaft von **Button 1** auf *false* gesetzt. Klickt er erneut auf **Button 3**, wird sie wieder auf *true* gesetzt.
- Während der Benutzer die linke Maustaste über **Button 4** gedrückt hält (*PreviewMouseLeftButtonDown*-Event), soll der Hintergrund (*Background*-Eigenschaft) von **Button 2** rot gezeichnet werden. Lässt er sie wieder los (*PreviewMouseLeftButtonUp*-Event), soll der Hintergrund von **Button 2** grün gezeichnet werden.

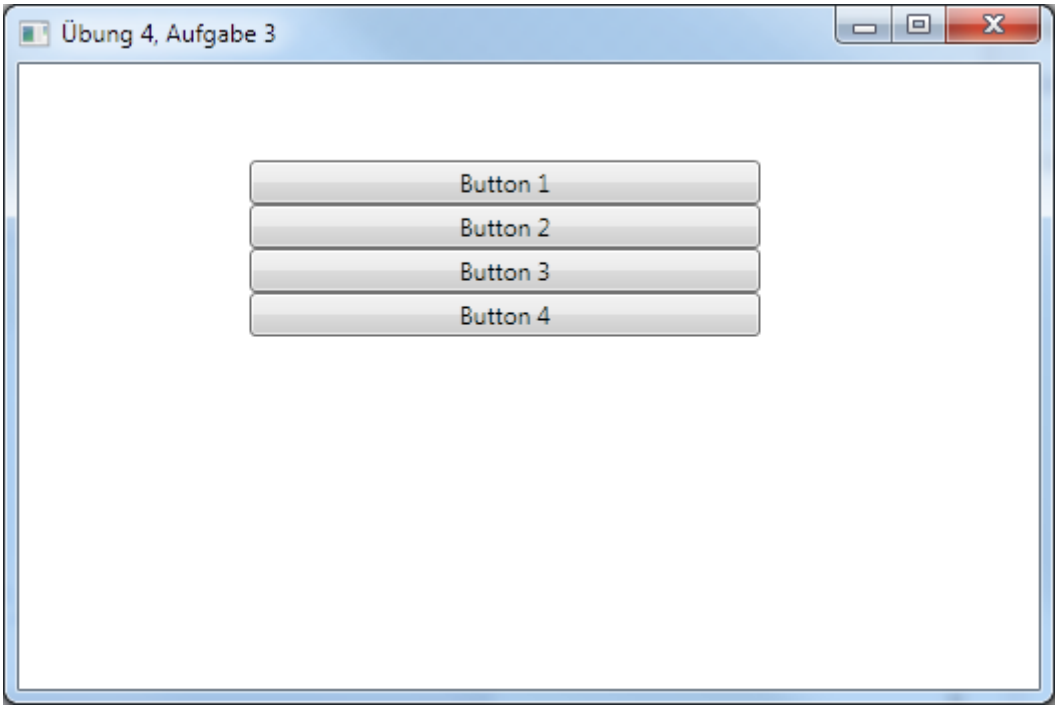


Abbildung 1