**WPF & MVVM Architechture**

Root UI is in MainWindow.xaml

* x:Class - corresponding .cs file (**partial** class MainWindow in .cs, to be combined with XAML)
  + InitializeComponent() 🡪 Combine with XAMl to create the MainWindow
* xmlns - XML namespace (XAML namespace is extension of XML) 🡪 Defines the namespace in which to resolve element names. xmlns is the default namespace by which an XML element name should be resolved.
  + xmlns:x - defines a **qualified** namespace for XAML specific elements.
  + xmlns:d - designer (display in Visual Studio)
  + xmlns:mc - markup compatibility
  + xmlns:views = "clr-namespace:MVVMDemo.Views" at the top - allow reference to user-defined components (or other .xaml views)
    - In code: <views:StudentView x:Name = “huhhhhh”… />
      * Meaning refer to StudentView class in MVVMDemo.Views namespace
      * If StudentView is a partial class that corresponds to a .xaml 🡪 **linked!**

\*\*Accessing Properties of GUI Elements from .cs code through its names

For example, we have a <Button x:Name = “myButton” Content = “hello”></Button>

We can access it using myButton.FontSize = 50;

**\*\***Property: **DataContext** **Class/Object** that will provide data to the current element (can be StudentView that directly makes up the view)

After setting DataContext, any bindings can bind using relative source reference

------------------------------------------------------------------------------------------------------------------------------------------

EXAMPLE: Model (Student class); ViewModel (StudentViewModel class); View (StudentView class)

StudentViewModel studentViewModel\_obj = new(); // created somewhere, for initializing

huhhhhh.DataContext = studentViewModel\_obj; 🡪 this object is data context for the XAML

<TextBox Text = “{Binding Path = studentViewModel\_obj.Students, Mode = TwoWay}” />

Ways to Hook up Views to ViewModel

1. **View First Construction in XAML** (calls the default constructor or the ViewModel while parsing the window. Makes use of setter for DataContext

<UserControl.DataContext>

<viewModel:StudentViewModel/>

</UserControl.DataContext>

<Grid> …..

1. **View First Construction in Code-behind** (like above)

App.xaml is used to include Styles as ResourceDictionary

A screen shot of a computer code

Description automatically generated with low confidence

A screen shot of a computer

Description automatically generated with low confidence**Grid**:

A screen shot of a computer code

Description automatically generated with low confidencePut elements in a Grid through Property Grid.Row/Column = “2”

If you set some rows to Auto and some to \*, the Auto rows will take up only the space required to fit their contents, and the remaining space will be divided up among the \* rows.

**GridSplitter:** Automatically used for resizing grids on both sides.

**Modularity:** For each portion of the grid, we can refer to a different view