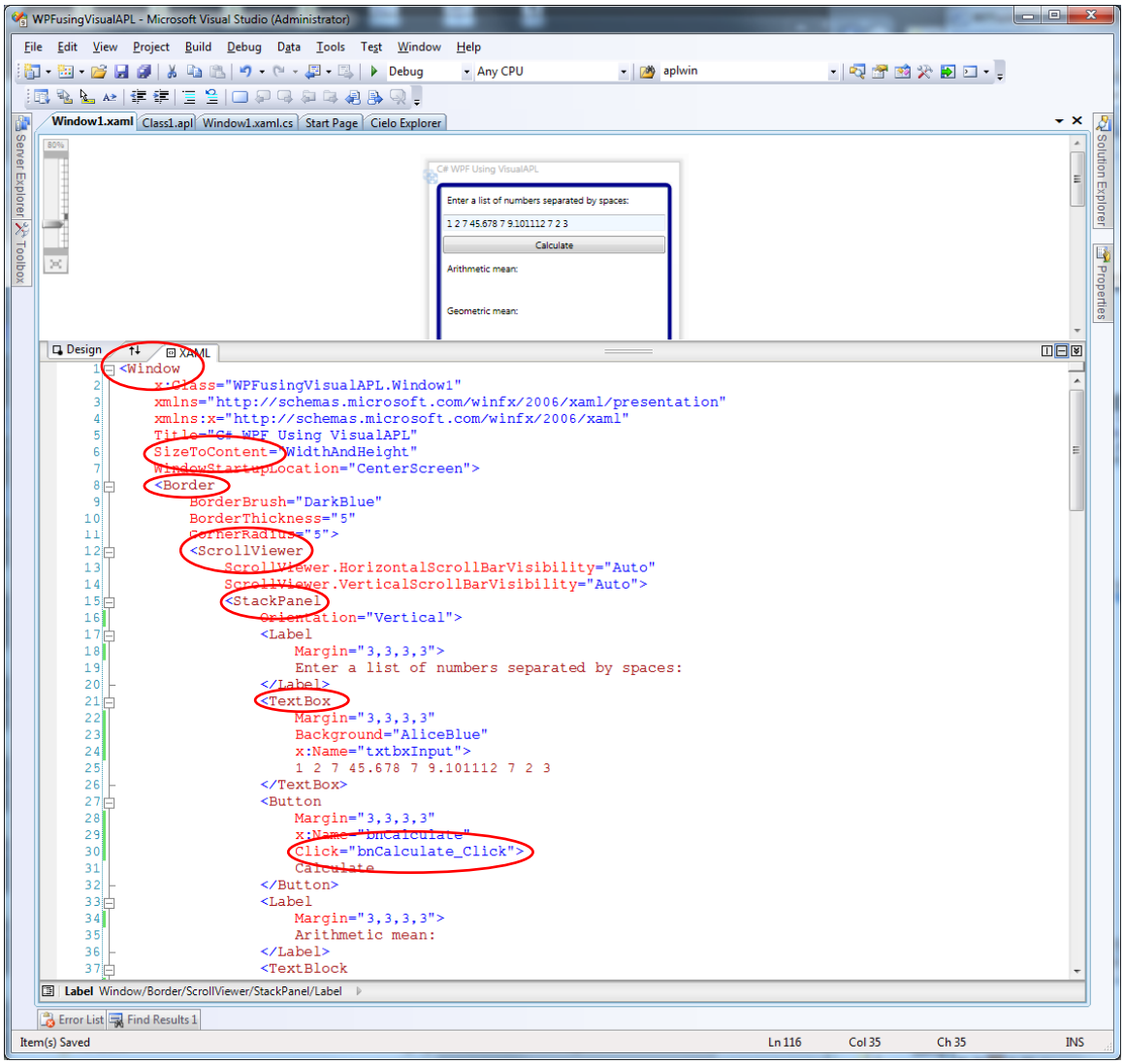
C# Windows Presentation Foundation Graphical User Interface Supported by a VisualAPL Business Rules Component

In this example a C# project which defines a Windows Presentation Foundation (WPF) graphical user interface (GUI) captures user-input data which is validated and processed by a business rules component built with VisualAPL. Both the C# WPF GUI and the VisualAPL components are a fully-managed .Net assemblies.

The WPF GUI is a window containing:

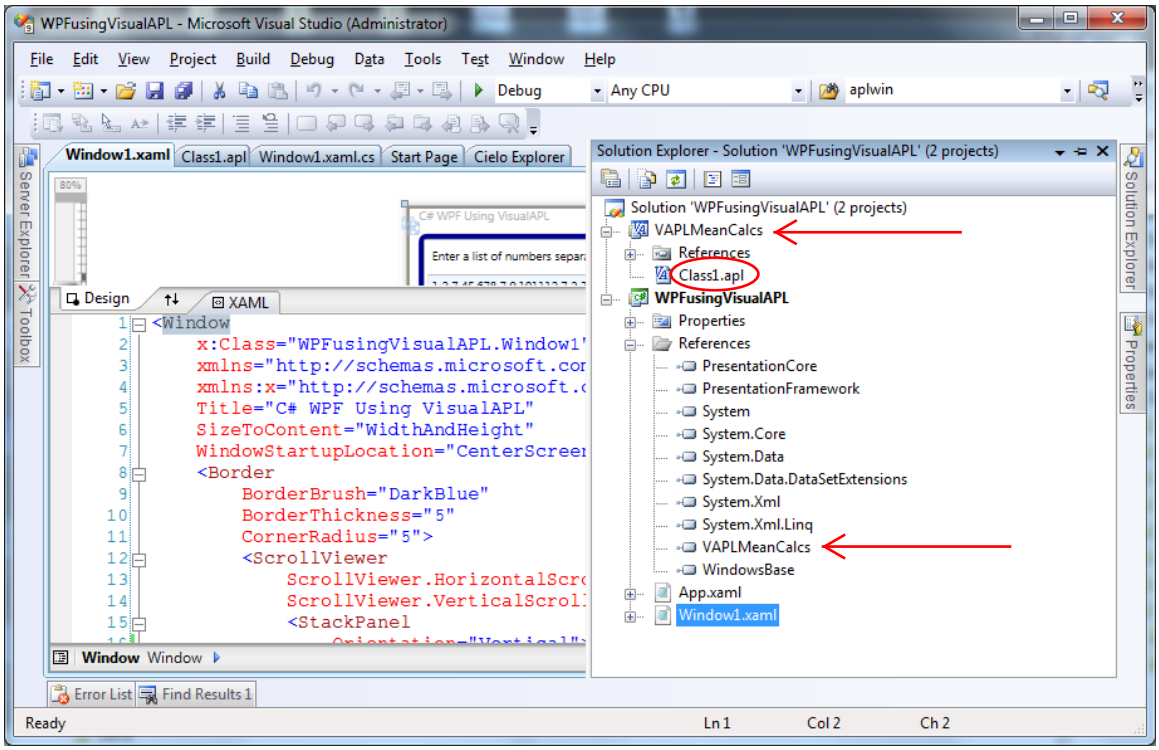
- An TextBox control to receive the user input
- A Button control which, when user-clicked, will run the VisualAPL functions that calculate the values for this application system
- Several TextBlock controls to contain the calculated results
- Several Label controls to identify the other control purposes to the user
- These controls are contained in a vertically-oriented StackPanel ‘container’ control
- The StackPanel control is contained in a ScrollViewer ‘container’ control with automatically-visible horizontal and vertical scrollbars
- The ScrollViewer control is contained in a Border ‘container’ control
- The Border control is contained in the top-most Window ‘container’ control of the application system
- The Window control employs the ‘SizeToContent’ attached property so that the entire GUI layout is automatically sized.

The C# WPF GUI is specified using an xml-format XAML file, “Window1.xaml”. VisualStudio renders the XAML file specification to visible controls on the design surface of the project.

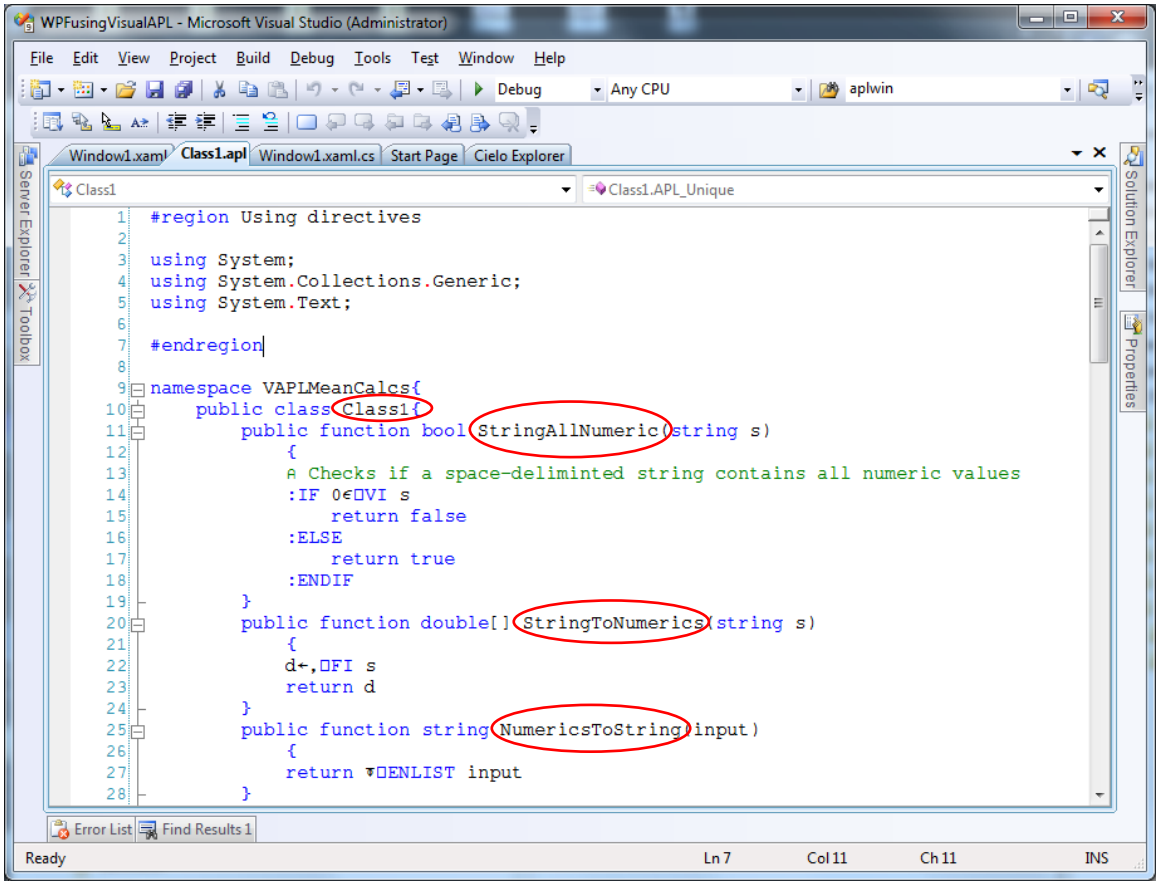

WPFusingVisualAPL - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Data Tools Test Window Help
Debug
Any CPU
aplwin
Window1.xaml
Class1.apl
Window1.xaml.cs
Start Page
Cielo Explorer
C# WPF Using VisualAPL
Enter a list of numbers separated by spaces:
1 2 7 45.678 7 9.101112 7 2 3
Calculate
Arithmetic mean:
Geometric mean:
Design
XAML
<Window
x:Class="WPFusingVisualAPL.Window1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="C# WPF Using VisualAPL"
SizeToContent="WidthAndHeight"
WindowStartupLocation="CenterScreen">
<Border
BorderBrush="DarkBlue"
BorderThickness="5"
CornerRadius="5">
<ScrollViewer
ScrollViewer.HorizontalScrollBarVisibility="Auto"
ScrollViewer.VerticalScrollBarVisibility="Auto">
<StackPanel
Orientation="Vertical">
<Label
Margin="3,3,3,3">
Enter a list of numbers separated by spaces:
</Label>
<TextBox
Margin="3,3,3,3"
Background="AliceBlue"
x:Name="txtbxInput">
1 2 7 45.678 7 9.101112 7 2 3
</TextBox>
<Button
Margin="3,3,3,3"
x:Name="bnCalculate"
Click="bnCalculate_Click">
Calculate
</Button>
<Label
Margin="3,3,3,3">
Arithmetic mean:
</Label>
<TextBlock
Label Window/Border/ScrollViewer/StackPanel/Label
Error List
Find Results 1
Item(s) Saved
Ln 116
Col 35
Ch 35
INS

The VisualAPL project, “VAPLMeanCalcs”, in this .Net solution contains a single class, “Class1” which contains all the VisualAPL methods and functions used to support the “business rules” component of the application system.

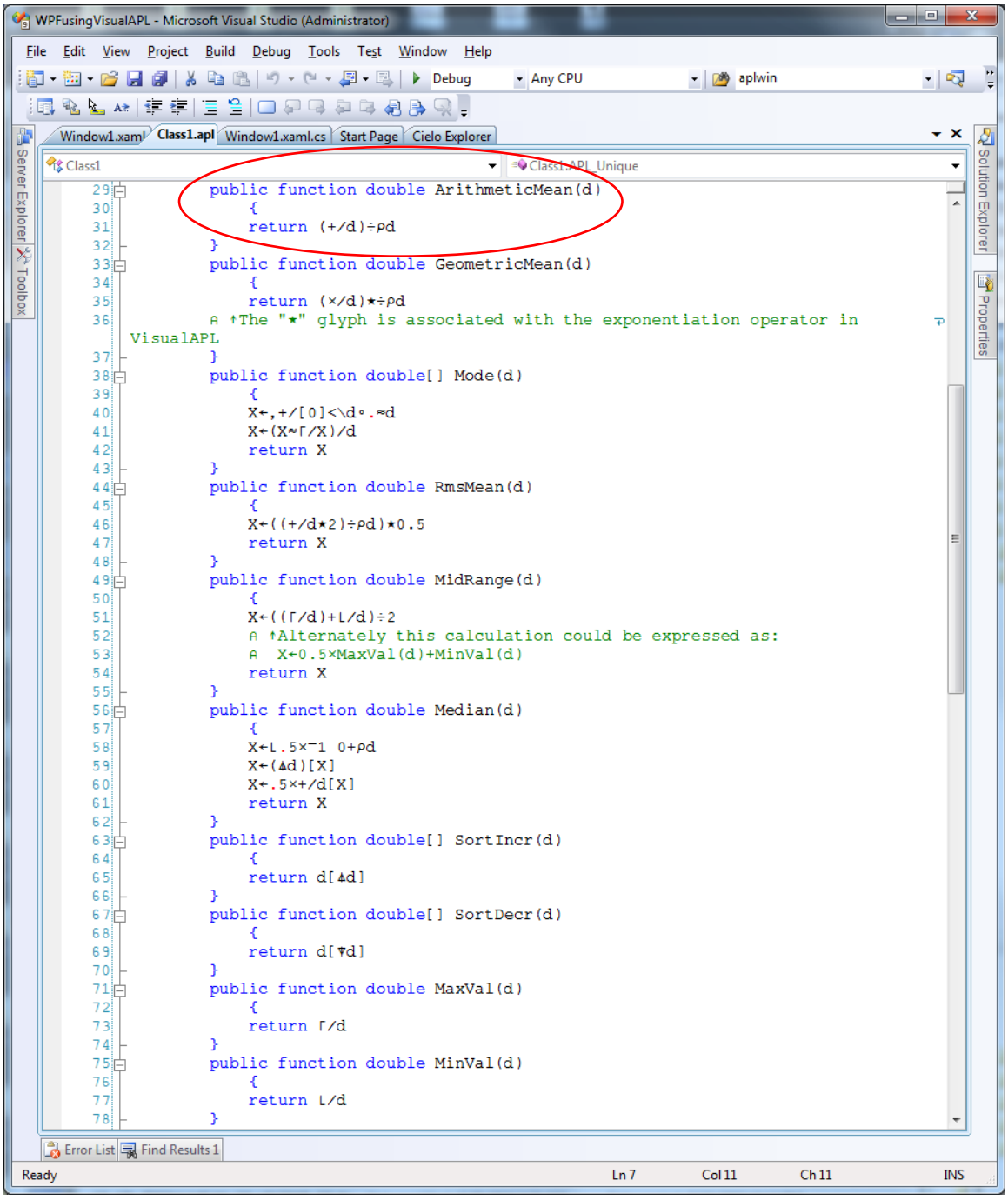
The C# WPF project in this .Net solution references the VisualAPL project so that an instance of the VisualAPL class can be created and used in the C# WPF project.



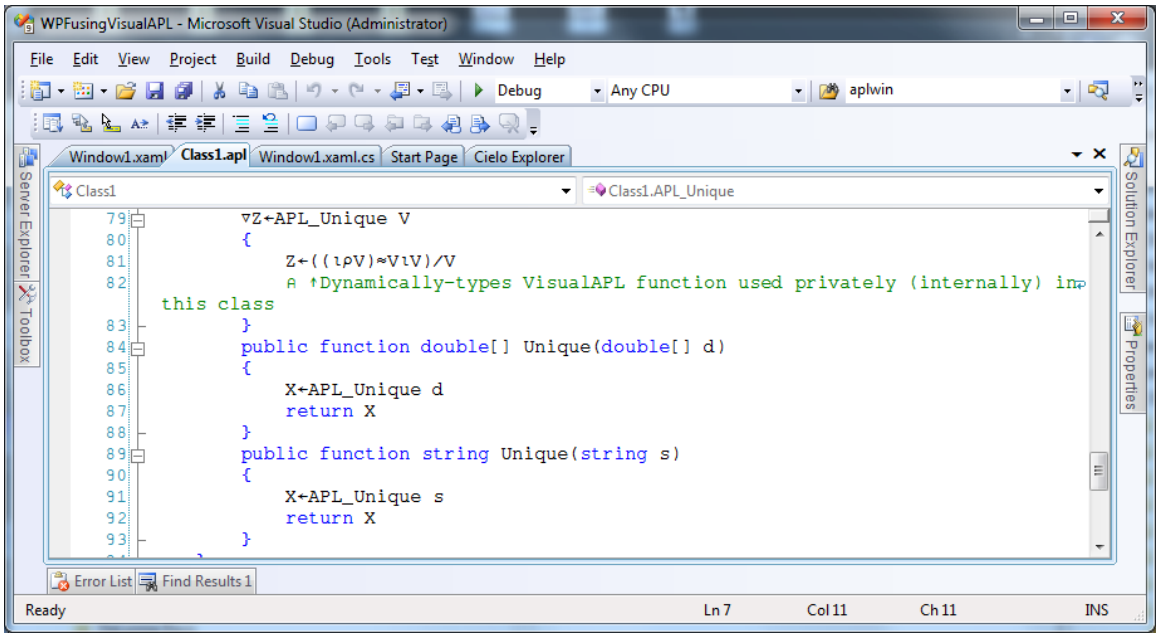
The VisualAPL “Class1” class contains several utility methods that will be used to validate the user-input data, convert the user-input data from string to numeric type and format the calculated values for illustration in the C# WPF GUI.



The VisualAPL “Class1” class also contains methods to compute the application-specific values from the user input collected by the C# WPF GUI and passed as arguments to the VisualAPL methods. These methods are all ‘public’ so that they will be exposed to the C# WPF GUI. In addition the function header syntax of these methods use the .Net standard format, i.e. “public function result_type function_name(argument_list){…}”, because this format is required for transparent interoperability between .Net assemblies.

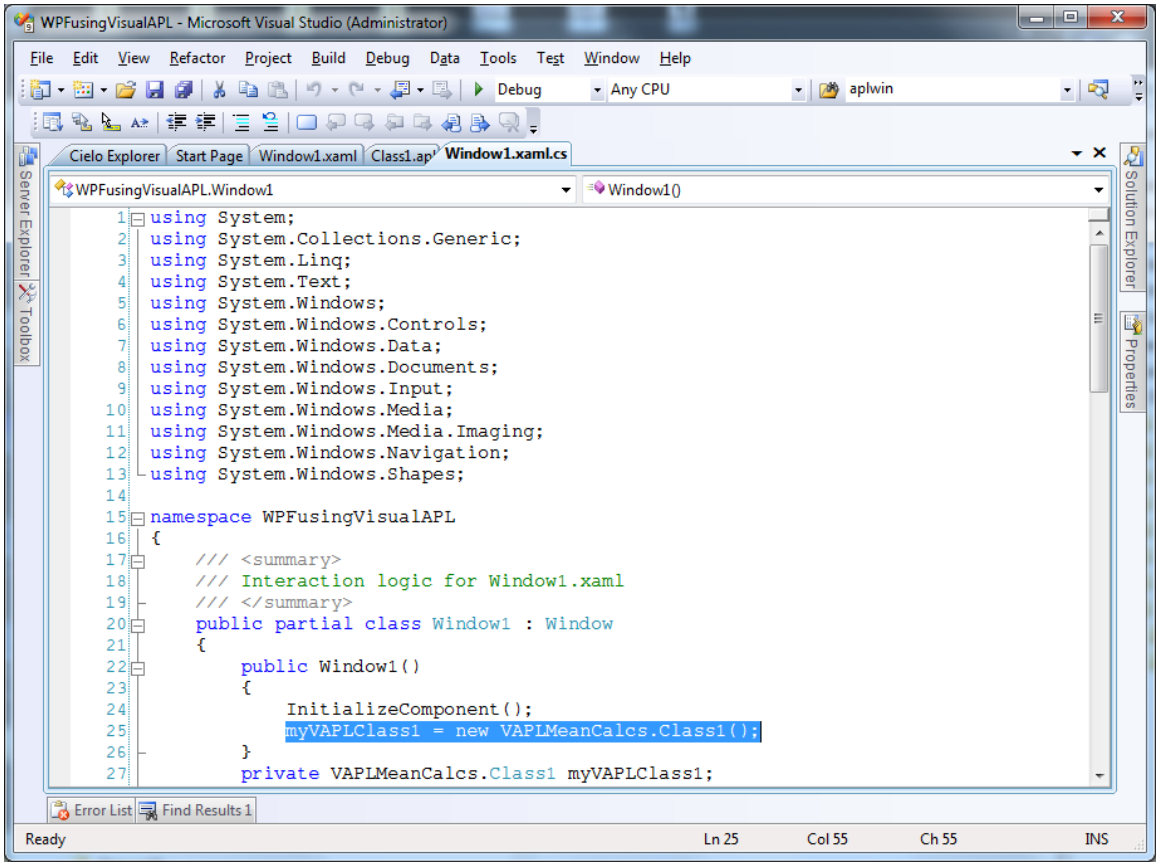


In the VisualAPL “Class1” class the implementation of the “Unique” method illustrates how several ‘overloads’ of the same function name can be created differing only in argument composition, e.g. number and data type of the arguments. The action of these overloaded “Unique” methods is supported by a single, dynamically-typed, traditional-format, VisualAPL function, “APL_Unique”.

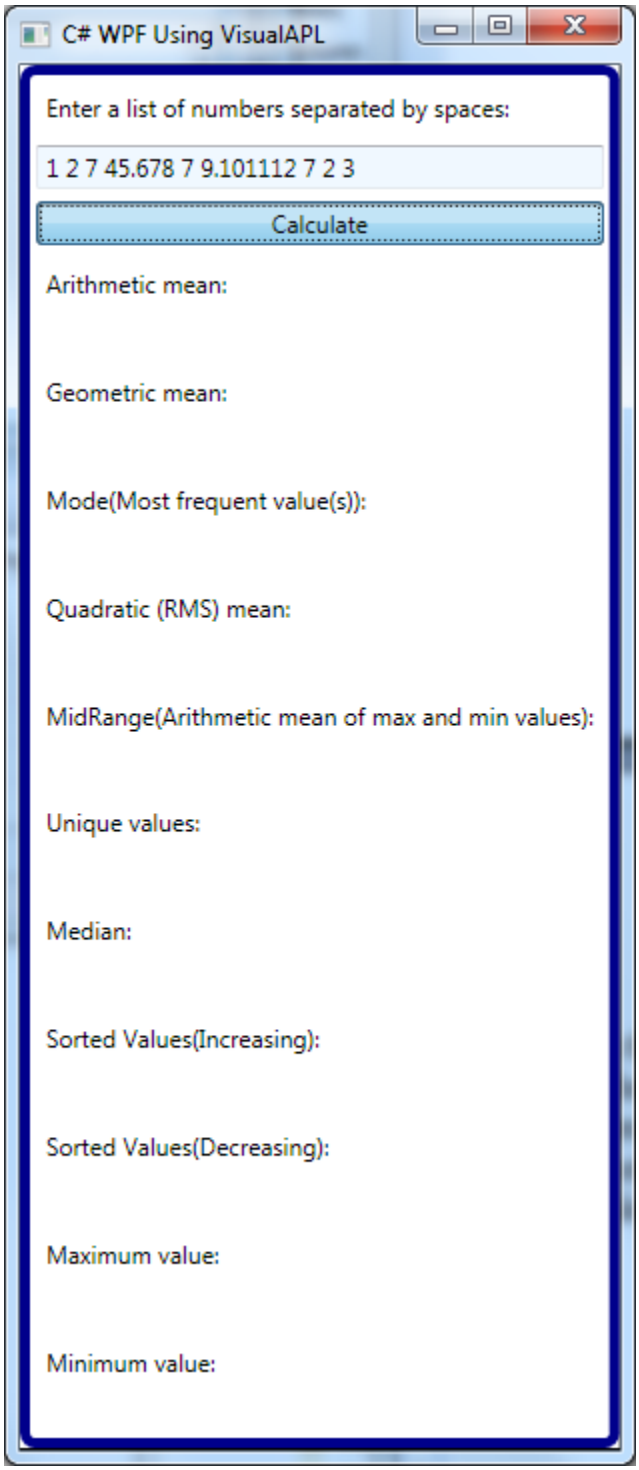


The C# WPF GUI project includes a ‘code-behind’ file containing the C# source code necessary to control application system interactions between the user and the GUI and between the GUI and the VisualAPL business rules component.

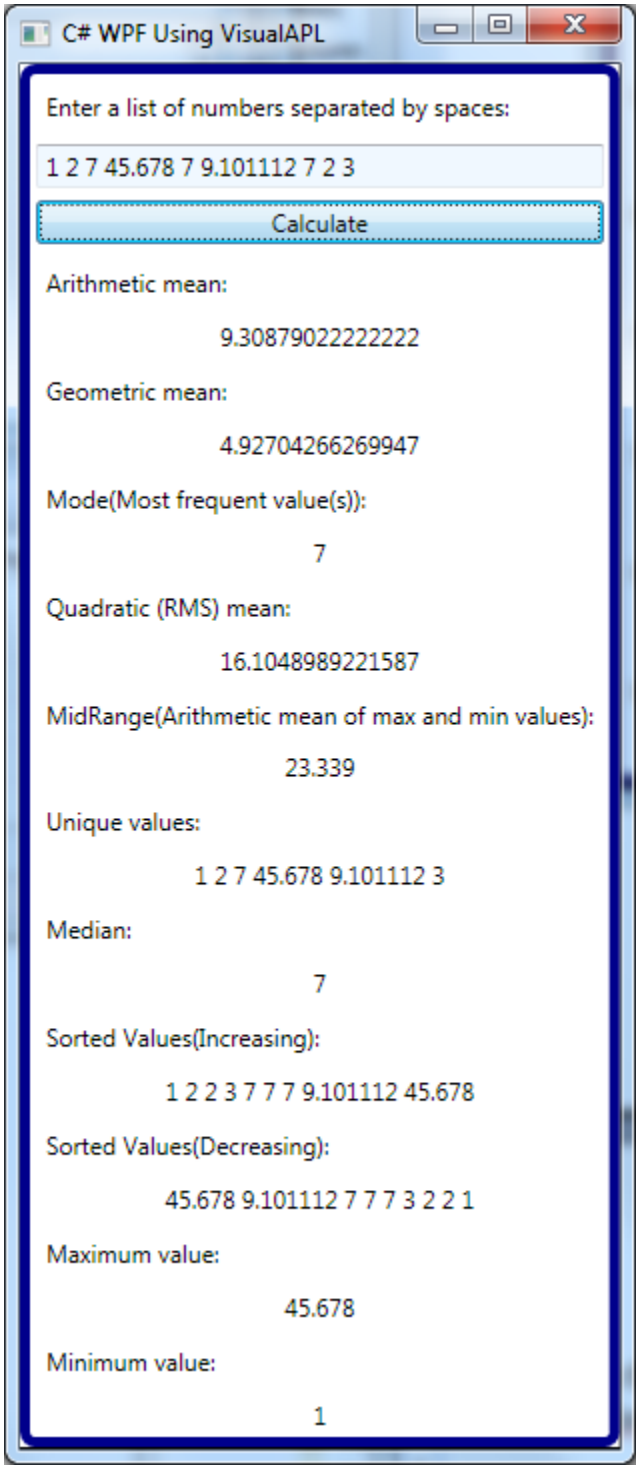
When the C# WPF GUI is initialized, an instance of the VisualAPL “Class1” class is created for use in the application system.



In this simple application system the interaction between the user and the GUI is done when the user enters the required input data and clicks the ‘Calculate’ button to request computation of the application-specific values based on that user input.



In this application system the interaction between the C# WPF GUI component and the VisualAPL business rules component is via the button click event handler method.



The C# WPF GUI code behind file, “Window1.xaml.cs” contains the button click event handler. The processing defined in this event handler occurs when the user clicks the ‘Calculate’ button control in the C# WPF GUI.

The processing involves:

- Disabling the GUI while the event handler processing is being performed
- Validation of the user input for no value using C#
- Validation of the user input for all numeric values using the VisualAPL “StringAllNumeric” method
- Conversion of the user input data from a string to an array of doubles using the VisualAPL “StringToNumerics” method
- Computation of the various application-specific values. C# passes the numeric user input “data” to the applicable VisualAPL method which returns the computed value
- Insertion of the computed values, converted to string, into the C# WPF GUI so it can be seen by the user
- Enabling the GUI after the event handler processing is complete