Department of Computer Science

08120 Programming 2
Week 29 Practical 2012/2013
Crazy Timers in Windows Presentation Foundation

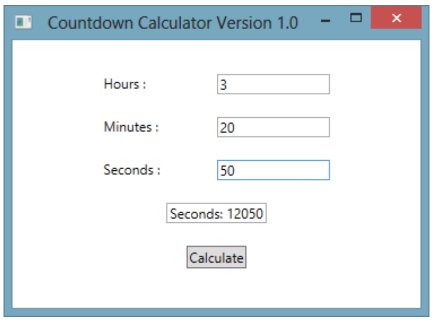
Last week you made a program for a company called "Crazy Timers". It converted a time in hours, minutes and seconds into a number of seconds. You have been employed to do some work on this program to make it into a Windows Presentation Foundation (WPF) application.

WPF Countdown calculator

Last time you made a program to work out the number of seconds there are in a time expressed in hours, minutes and seconds. The user enters the number of hours, minute and seconds and the program then displays the number of seconds equivalent to that time. The program was used in the following way:

```
Countdown Timer Calculator by Rob Miles
Version 1.0
Enter the number of hours: 3
Enter the number of minutes: 20
Enter the number of seconds: 50
The total number of seconds is: 12050
```

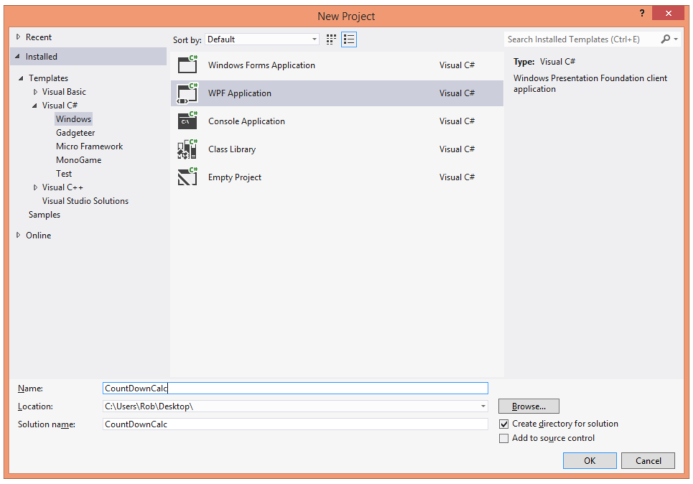
Now they want a program with a Windows Presentation Foundation (WPF) interface that works in the same way:


Countdown Calculator Version 1.0
Hours : 3
Minutes : 20
Seconds : 50
Seconds: 12050
Calculate

The user types in the numbers and presses the calculate button. The answer is displayed on the form. You can use some of the program code from your previous solution, but you will have to write some new code as well. You will use Visual Studio to create the application.

Now you can create the program itself:

1. Log in and start up Visual Studio 2012
2. Use **File>New>Project** to open the New Project dialogue. The dialog you see might not exactly match the one below, but it will have a Visual C# item which contains a WPF (Windows Presentation Foundation) Application template. Be careful not to create a Visual Basic project as this would be very confusing.



Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
Class Library	Visual C#
Empty Project	Visual C#

Type: Visual C#
Windows Presentation Foundation client application

Name: CountDownCalc
Location: C:\Users\Rob\Desktop\
Solution name: CountDownCalc
[✓] Create directory for solution
[ ] Add to source control

You are creating a Windows Forms application called **CountDownCalc**.

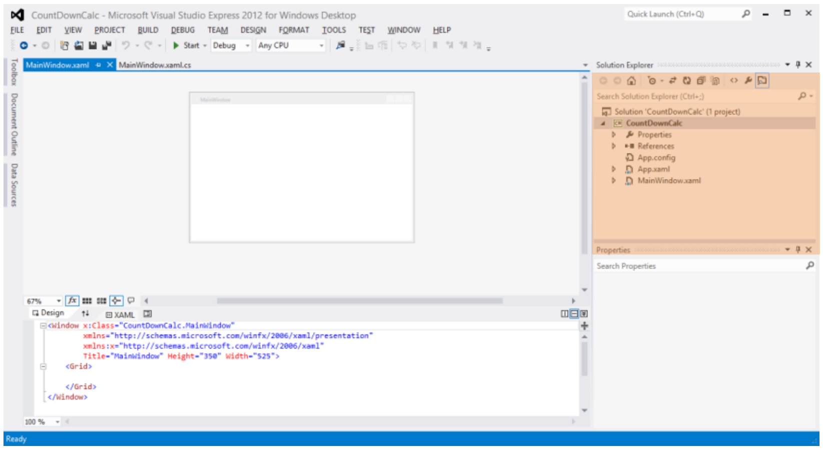
Before you go any further; perform the following:

3. Select the template as above, give the project the name **CountDownCalc** and click OK. Visual Studio will create a new solution containing a Windows Presentation Foundation application.

Designing a WPF Application using Visual Studio

Visual Studio is a very complex piece of software which can be a bit hard to find your way around. Here are some of the more important areas. Note that your window arrangement might not look like the ones below. You can use **Window>Reset Window Layout** to put them back to a similar appearance to the ones below if you get stuck.

Finding a Windows Design using Solution Explorer

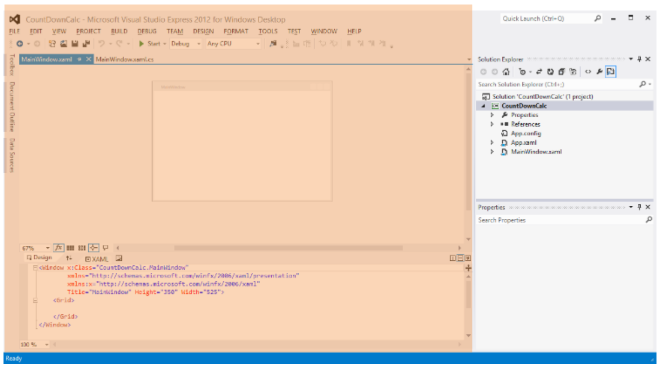


```
<Window x:Class="CountDownCalc.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>

    </Grid>
</Window>
```

All the projects are shown in this area. You select the one you want to work on by double clicking on it. If you want to edit the main window of your application, you can double click on **MainWindow.xaml** in the Solution Explorer window.

Remember that you can have two views of a Window in your program. You can look at how it will appear to the user, with the buttons and other components you have added, and you can look at the XAML code which describes all the items on the window.

Working on the Design of a Window



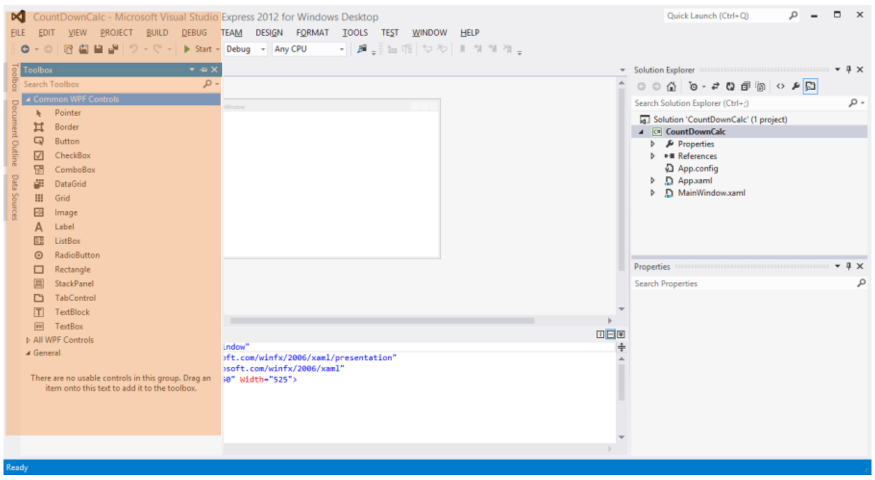
```
<Window x:Class="CountDownCalc.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>

    </Grid>
</Window>
```

This is where you work on the code or design of your program.

When the project is created it opens the main page for your application. This shows the layout of the page and the XAML that describes it. You can edit the appearance of the form by adding to the XAML or by using the toolbox to add components to the page.

We are going to start by using the Toolbox to obtain some elements and then work on their properties. Later you might want to investigate using the XAML text to adjust elements on the page. Sometimes this is quicker and more precise that trying to use the mouse.

Adding Components using the Toolbox



The Toolbox is where you get your components from when you want to add them to your window. If the Toolbox is not visible you can make it appear by selecting **View>Other Windows>Toolbox**

You add components to a window design by selecting the component in the toolbox and then dragging it onto the window. You can use the mouse to position the component and set its size.

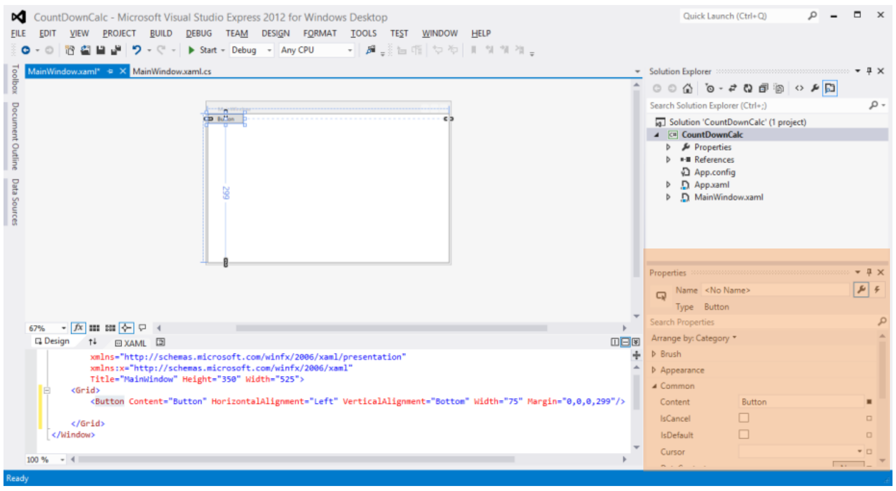
While the Toolbox provides a nice user interface that allows you to “Drag and Drop” items onto the window the actual layout of the window is expressed by the text in the XAML (eXtensible Application Markup Language) file.

If you add something using the Toolbox you will notice that Visual Studio will automatically create and update the XAML text that describes the components on the window. If you move the components or change their properties the XAML code is updated to keep track of the changes. The components that you will need for the countdown timer are Label (just put text on the page), TextBox (somewhere the user can type their inputs) and Button (used to trigger an action).

It is perfectly possible to create components just by adding lines to the XAML rather than using the Toolbox. This can be useful if you have to create a large number of items, you can just copy the text in the XAML file and edit it by hand. It is worth spending some time working out how XAML is constructed, this gives you a great insight into how your program display is created.

Properties

Everything that Visual Studio manages has *properties*. You can regard properties as settings and information about an item.



```
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Button Content="Button" HorizontalAlignment="Left" VerticalAlignment="Bottom" Width="75" Margin="0,0,0,299"/>

    </Grid>
</Window>
```

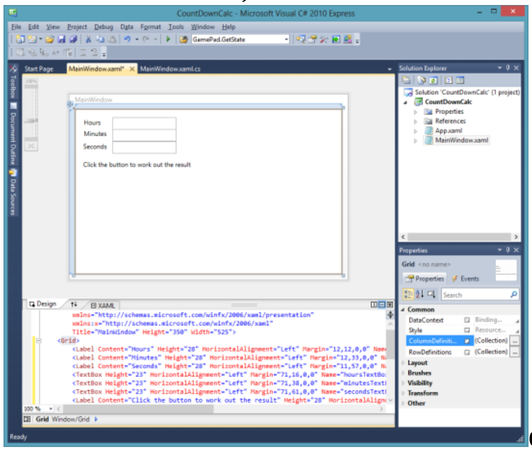
A project will have properties such as the name of the project and where it is stored. A Label will have properties such as the name of the label, its position and size, the text in the label, how big the label is, the colour of the background and so on.

A button will have properties such as the text on the button, the colour of the button, its position on the screen and so on.

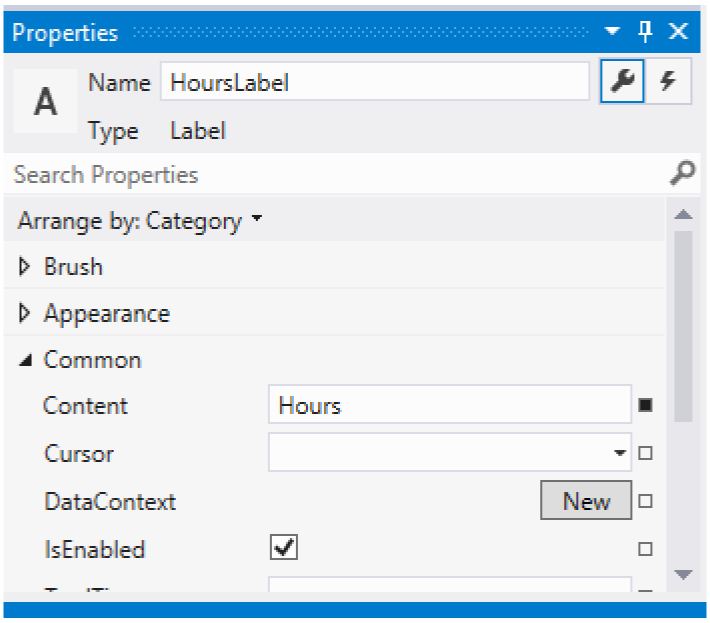
The Property pane is where these are managed. The contents of this change depending on what is presently selected in the editor. You can edit the properties to change the information about that particular thing. When you add a Label to the form you will use the properties area to set the variable name of the Label in the program and the text that the Label displays. You can open the properties box for any item by right-clicking it on the design surface and then selecting "Properties" from the window that appears. Of course, if you change settings in the property Visual Studio will reflect these changes by updating the values in the XAML.

Building the Window

Now you are going to build up the user interface that you saw at the start of this lab. The components you are going to use are **Label**, **TextBox** and **Button**. The Label will be used to convey information to the user, a TextBox lets the user type something in and a Button can generate an event and make the program do something in response to the button click.



```
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Label Content="Hours" Height="28" HorizontalAlignment="Left" Margin="12,12,0,0" Nam
        <Label Content="Minutes" Height="28" HorizontalAlignment="Left" Margin="12,33,0,0" N
        <Label Content="Seconds" Height="28" HorizontalAlignment="Left" Margin="11,57,0,0" N
        <TextBox Height="23" HorizontalAlignment="Left" Margin="71,16,0,0" Name="hoursTextBo
        <TextBox Height="23" HorizontalAlignment="Left" Margin="71,38,0,0" Name="minutesText
        <TextBox Height="23" HorizontalAlignment="Left" Margin="71,61,0,0" Name="secondsText
        <Label Content="Click the button to work out the result" Height="28" HorizontalAlign
```


Properties

Name: HoursLabel

Type: Label

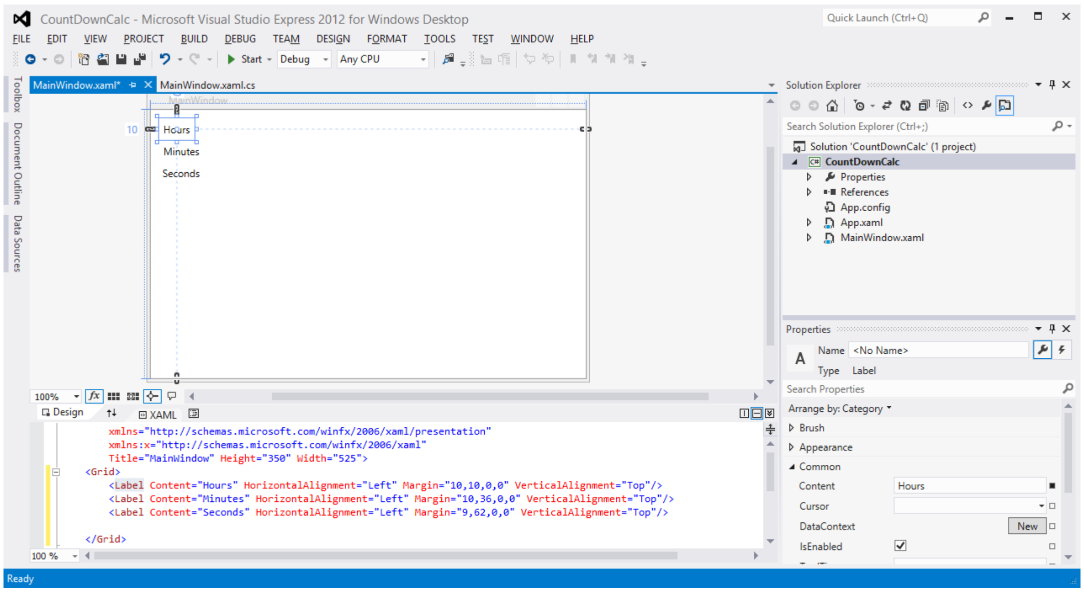
Arrange by: Category

- Brush
- Appearance
- Common

Content	Hours
Cursor	
DataContext	New
IsEnabled	✓

Before you go any further; perform the following:

1. Click on the **Label** item in the **Toobox** and drag it onto the window surface. This will be the label for the Hours entry for our program. Position the label in the top left hand corner.
2. Whilst the label is selected, go into the Properties part of Visual Studio and set the **Content** of the Label to "**Hours**", as shown above.
3. Create two other Label items with content "**Minutes**" and "**Seconds**" respectively. This should leave you with a display that looks like the one below.



CountDownCalc - Microsoft Visual Studio Express 2012 for Windows Desktop
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM DESIGN FORMAT TOOLS TEST WINDOW HELP
MainWindow.xaml*
MainWindow.xaml.cs
Hours
Minutes
Seconds
Solution Explorer
Solution 'CountDownCalc' (1 project)
CountDownCalc
Properties
References
App.config
App.xaml
MainWindow.xaml
Ready

```
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Label Content="Hours" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top"/>
        <Label Content="Minutes" HorizontalAlignment="Left" Margin="10,36,0,0" VerticalAlignment="Top"/>
        <Label Content="Seconds" HorizontalAlignment="Left" Margin="9,62,0,0" VerticalAlignment="Top"/>

    </Grid>
```

Properties

Name: <No Name>

Type: Label

Content	Hours
Cursor	
DataContext	New
IsEnabled	✓

These labels will tell the user where to type in the numbers to be used by the program. Next you need to add the textboxes which will hold the values that the user will type in.

Before you go any further; perform the following:

1. Create three **TextBox** items with the names **hoursTextBox**, **minsTextBox** and **secondsTextBox**. These should be lined up against their respective labels. **Note that we have to give these items names, as the program will be referring to them when it runs.**
2. You will also need to add a **Label** which will be used to display the result. Give this label the name **resultLabel** and set the text it displays to "Click Calculate to work out result"

At this point you should have all the elements you need to make your program work.

The XAML Markup File

It is worth looking at the XAM to see how this all fits together. The top part of the XAML file identifies the XML “schema” (the definition of what can appear in a XAML page) and some system libraries associated with the page, along with the title of the page and other properties.

The main element on the page is the **<Grid>** container which can hold a number of items which are positioned in the grid by setting the Margin value for each of them. The XMAL for the page that I created is as follows:

```
<Grid>
    <Label Content="Hours" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top"/>
    <Label Content="Minutes" HorizontalAlignment="Left" Margin="10,36,0,0" VerticalAlignment="Top"/>
    <Label Content="Seconds" HorizontalAlignment="Left" Margin="9,62,0,0" VerticalAlignment="Top"/>
    <TextBox x:Name="HoursTextBox" HorizontalAlignment="Left" Height="23" Margin="87,10,0,0"
     TextWrapping="Wrap" Text="TextBox" VerticalAlignment="Top" Width="120"/>
    <TextBox x:Name="minutesTextBox" HorizontalAlignment="Left" Height="23" Margin="87,36,0,0"
     TextWrapping="Wrap" Text="TextBox" VerticalAlignment="Top" Width="120"/>
    <TextBox x:Name="SecondsTextBox" HorizontalAlignment="Left" Height="23" Margin="87,62,0,0"
     TextWrapping="Wrap" Text="TextBox" VerticalAlignment="Top" Width="120"/>
    <Label x:Name="ResultLabel" Content="Click Calculate to work out the result"
     HorizontalAlignment="Left" Margin="45,93,0,0" VerticalAlignment="Top"/>
</Grid>
```

It is interesting that if I change the content of this file the display produced by the program will also change. You will also see such changes reflected in the visual editor provided by Visual Studio.

Before you go any further; perform the following:

1. Select the text view of the XAML file for MainWindow.xaml.
2. Change the text for **ResultLabel** to **“Please click the button to work out the result”**.
3. Note that the display in the editor changes, as do the contents of the properties for this item.

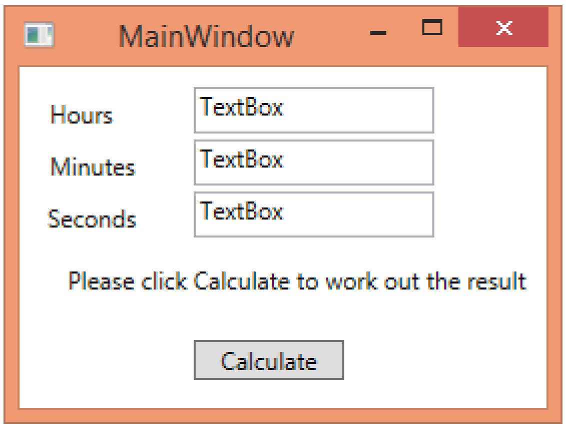
Adding the Button

Next you need to add the Calculate button which the user will press when they want to calculate the result.

Before you go any further; perform the following:

4. Click on the **Button** item on the Toolbox and drag it onto the Form in the bottom right hand corner. Change the **Name** of the Button to **calculateButton** and the **Content** on the button to **Calculate.**

If you run the program again now you will have a form which contains a button you can click, but at the moment it doesn't do anything.



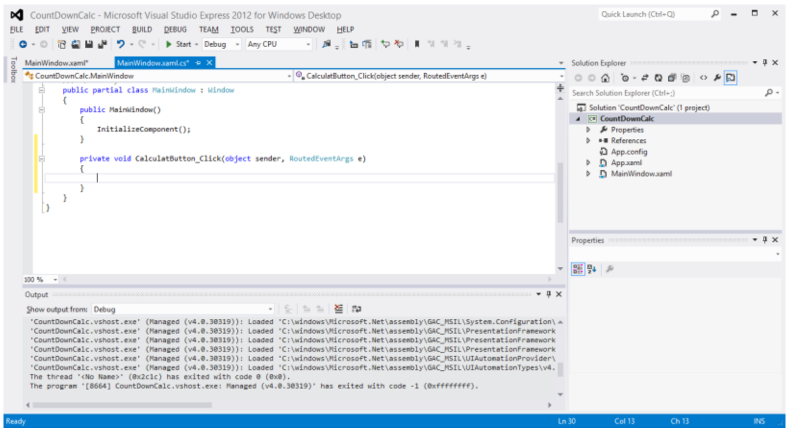
I've also re-sized the window in this example to make it look better. You can stop the program by closing the window using the X in the top right hand corner, as for any other Windows application.

Handing the button clicked event

The next thing is to bind a method to **CalculateButton** which runs when the buttons is clicked. This is very easy to do.

Before you go any further; perform the following:

5. In the Visual Studio designer, double click on the **calculateButton** on the form. If you do this right Visual Studio will create an event handler for the **Click** event, bind a method to that event handler and take you to that method in the program page linked to the XAML design. The screen should change to the one you see below.


CountDownCalc - Microsoft Visual Studio Express 2012 for Windows Desktop
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST WINDOW HELP
Start
Debug
Any CPU
MainWindow.xaml*
MainWindow.xaml.cs*
CountDownCalc.MainWindow
CalculatButton_Click(object sender, RoutedEventArgs e)
Solution Explorer
Solution 'CountDownCalc' (1 project)
CountDownCalc
Properties
References
App.config
App.xaml
MainWindow.xaml
Properties
Output
Show output from: Debug
Ready
Ln 30
Col 13
Ch 13
INS

```
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void CalculatButton_Click(object sender, RoutedEventArgs e)
        {
            
        }
    }
}
```

```
'CountDownCalc.vshost.exe' (Managed (v4.0.30319)): Loaded 'C:\windows\Microsoft.Net\assembly\GAC_MSIL\System.Configuration\
'CountDownCalc.vshost.exe' (Managed (v4.0.30319)): Loaded 'C:\windows\Microsoft.Net\assembly\GAC_MSIL\PresentationFramework
'CountDownCalc.vshost.exe' (Managed (v4.0.30319)): Loaded 'C:\windows\Microsoft.Net\assembly\GAC_MSIL\PresentationFramework
'CountDownCalc.vshost.exe' (Managed (v4.0.30319)): Loaded 'C:\windows\Microsoft.Net\assembly\GAC_MSIL\PresentationFramework
'CountDownCalc.vshost.exe' (Managed (v4.0.30319)): Loaded 'C:\windows\Microsoft.Net\assembly\GAC_MSIL\UIAutomationProvider\
'CountDownCalc.vshost.exe' (Managed (v4.0.30319)): Loaded 'C:\windows\Microsoft.Net\assembly\GAC_MSIL\UIAutomationTypes\v4.
The thread '<No Name>' (0x2c1c) has exited with code 0 (0x0).
The program '[8664] CountDownCalc.vshost.exe: Managed (v4.0.30319)' has exited with code -1 (0xffffffff).
```

When **calculateButton** is clicked the method **calculateButton_Click** is called. This must take the text out of each **TextBox**, convert it into numbers, work out the number of seconds and then display that value in the result label. We could start by just setting the text of the result label to "Calculate Clicked".

Before you go any further; perform the following:

6. Add the statement:

```
resultLabel.Content = "Calculate Clicked";
```

   to the **calculateButton_Click** method.
7. Run the program and note that when you click the button the text in the label changes.

Performing the Calculation

Now you need to make the code in the event hander actually perform the calculation. You can get the string out of a TextBox by using the Text property of that TextBox:

```
    int seconds;
    seconds = int.Parse(secondsTextBox.Text);
```

This would convert the text in the **secondsTextBox** into an integer you can use to work out the answer.

Before you go any further; perform the following:

8. Create a method that works out the number of seconds and displays them on the form.

Adding Error Handling

At the moment the program will work, but it will not behave correctly if the user types invalid numbers into a **TextBox**, or leaves something out. You need to add error handling to make your program into a proper solution. The best way to do this is to catch exceptions (using the usual try-catch construction) and then use these to send messages to your users.

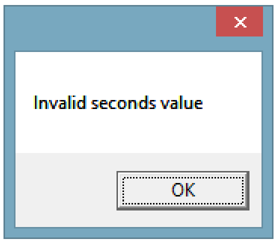
Text Colour

You could indicate that an entry is wrong by changing the colour to red. You can do this by changing the **Foreground** property of a **Label** or **TextBox** so that the text in it has a different colour:

```
secondsTextBox.Foreground = new SolidColorBrush(Colors.Red);
```

You can change to green or other colours in the same way. You can also change the background by using the **BackColor** property. Remember that the spellings here are all American.

Message Box


Invalid seconds value

You can make your program pop up a message box with a warning message in it. This will be displayed until the user clears the message.

You can get the message displayed by using the following:

```
    MessageBox.Show("Invalid Seconds Value");
```

The **MessageBox** is a class built into Windows (rather like the Console) which you can use to display quick messages. The program pauses in the Show method until the user presses the OK button, at which point it continues at the statement following the call of Show.

Before you go any further; perform the following:

1. Add error handling to your program so that it displays appropriate messages if the user does not enter valid values for the numbers.

2. Show a demonstrator the program working. They will want to see your program working and may have some tests for it.

Reverse Countdown

If you have time, create a brand new Windows application that performs the "reverse countdown" calculation and can be used to convert seconds back into hours, minutes and seconds.

Rob Miles
February 2014