```


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace Motoring
{
    /// <summary>
    /// Interaction logic for PageRunning.xaml
    /// </summary>
    public partial class PageRunning : Page
    {
        //-- variables to hold period values selected in the combos
        //-- Placing them here makes them global in the module
        //-- This is a different approach to previous used
        private string insurancePeriod = "Annual";     //-- Initialised Annual
        private string fuelPeriod = "Annual";          //-- Ditto
        private string servicingPeriod = "Annual";     //-- Ditto
        private string roadTaxPeriod = "Annual";       //-- Ditto
        //-- Numeric values
        private int insurance = 0;                //-- Initialised to 0
        private int fuel = 0;                     //-- Ditto
        private int servicing = 0;                //-- Ditto
        private int roadTax = 0;                  //-- Ditto

        //-- Combo items list of strings
        List<string> costPeriods = new List<string>();

        //-- Object to manage data brought forward
        CostSummary summaryBroughtForward = new CostSummary();

        public PageRunning(CostSummary summaryPassed)
        {
            InitializeComponent();
            summaryBroughtForward = summaryPassed;
            costPeriods = Periods();

            //-- Combo data sources (maybe redundant)
            InsurancePeriodCombo.ItemsSource = Periods();
            FuelPeriodCombo.ItemsSource = Periods();
            ServicingPeriodCombo.ItemsSource = Periods();
            RoadTaxPeriodCombo.ItemsSource = Periods();
        }

```

```csharp
        private void SummaryPageButton_OnClick(object sender, RoutedEventArgs e)
        {
            CostSummary summary = new CostSummary();

            summary = summaryBroughtForward;


            //-- Run the method to get the values in the form controls
            HarvestValues();

            //-- object to hold the RunningCost data
            //-- Initialise the object with the current values held in the module-wide variables
            RunningCost runningCost = new RunningCost
            {
                Insurance = insurance,
                InsurancePeriod = insurancePeriod,
                Fuel = fuel,
                FuelPeriod = fuelPeriod,
                Servicing = servicing,
                ServicingPeriod = servicingPeriod,
                RoadTax = roadTax,
                RoadTaxPeriod = roadTaxPeriod
            };

            summary.RunningCost = runningCost;

            var pageSummary = new PageSummary(summary);
            this.NavigationService.Navigate(pageSummary);

        }

        // -- Combo box data population
        List<string> Periods()
        {
            List<string> myList = new List<string>
            {
                "Annual",
                "Monthly",
                "Weekly"
            };

            return myList;
        }

        //-- Combo Box Load events
        private void InsurancePeriodCombo_OnLoaded(object sender, RoutedEventArgs e)
        {
            var combo = sender as ComboBox;
            combo.ItemsSource = costPeriods;
            combo.SelectedIndex = 0;
        }

        private void FuelPeriodCombo_OnLoaded(object sender, RoutedEventArgs
            e)
        {
            var combo = sender as ComboBox;
            combo.ItemsSource = costPeriods;
            combo.SelectedIndex = 0;
        }

        private void ServicingPeriodCombo_OnLoaded(object sender, RoutedEventArgs e)
        {
            var combo = sender as ComboBox;
            combo.ItemsSource = costPeriods;
            combo.SelectedIndex = 0;
        }

        private void RoadTaxPeriodCombo_OnLoaded(object sender, RoutedEventArgs e)
        {
            var combo = sender as ComboBox;
            combo.ItemsSource = costPeriods;
            combo.SelectedIndex = 0;
        }

        private void InsurancePeriodCombo_OnSelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            var selectedComboItem = sender as ComboBox;
            insurancePeriod = selectedComboItem.SelectedItem as string;
        }

        private void FuelPeriodCombo_OnSelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            var selectedComboItem = sender as ComboBox;
            fuelPeriod = selectedComboItem.SelectedItem as string;
        }

        private void ServicingPeriodCombo_OnSelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            var selectedComboItem = sender as ComboBox;
            servicingPeriod = selectedComboItem.SelectedItem as string;
        }

        private void RoadTaxPeriodCombo_OnSelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            var selectedComboItem = sender as ComboBox;
            roadTaxPeriod = selectedComboItem.SelectedItem as string;
        }

        //-- Harvest numeric values from the Text Boxes
        private void HarvestValues()
        {
            //-- Insurance
            if (InsuranceCostTextBox.Text != "")
```

```csharp
            {
                if (int.TryParse(InsuranceCostTextBox.Text, out int insuranceValue))
                {
                    insurance = insuranceValue;
                }
                else
                {
                    MessageBox.Show("The insurance cost must be a whole number");
                }
            }
            else
            {
                MessageBox.Show("Enter the insurance cost, even if it's 0");
            }

            //-- Fuel
            if (FuelCostTextBox.Text != "")
            {
                if (int.TryParse(FuelCostTextBox.Text, out int fuelValue))
                {
                    fuel = fuelValue;
                }
                else
                {
                    MessageBox.Show("The fuel cost must be a whole number");
                }
            }
            else
            {
                MessageBox.Show("You must enter a fuel cost, evem if it's 0");
            }

            //-- Servicing
            if (ServicingCostTextBox.Text != "")
            {
                if (int.TryParse(ServicingCostTextBox.Text, out int servicingValue))
                {
                    servicing = servicingValue;
                }
                else
                {
                    MessageBox.Show("The servicing cost must be a whole number, even if it's 0");
                }
            }
            else
            {
                MessageBox.Show("You must enter a servicing cost, even if it's 0");
            }

            //-- Road tax
            if (RoadTaxCostTextBox.Text != "")
```

```
                {
                    if (int.TryParse(RoadTaxCostTextBox.Text, out int 
                      roadTaxValue))
                    {
                        roadTax = roadTaxValue;
                    }
                    else
                    {
                        MessageBox.Show("The road tax amount must be a whole 
                       number, even if it's 0");
                    }
                }
                else
                {
                    MessageBox.Show("you must enter a Road Tax value, even it it's 
                       0");
                }

            }
        }
    }
```