```
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;

/*  TITLE:      PageRunning
 *  AUTHOR:     Paul McKillop
 *  DATE:       October 2022
 *  PURPOSE:    PageRunning interaction code
 */

namespace McKillopMotoring
{
    /// <summary>
    /// Interaction logic for PageRunning.xaml
    /// </summary>
    public partial class PageRunning : Page
    {
        //-- variables to hold period values selected in the combos
        //-- Placing them here allows use throughout 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();
            // -- Debug message
            // -- TODO:: Remove message on build
            // string price = summaryPassed.CurrentLoan.CarPrice.ToString();
            // MessageBox.Show("The Car Price passed from PageLoan was " + price);

            summaryBroughtForward = summaryPassed;
            costPeriods = Periods();

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

```csharp
            ServicingPeriodCombo.ItemsSource = Periods();
            RoadTaxPeriodCombo.ItemsSource = Periods();


        }

        private void ClearButton_Click(object sender, RoutedEventArgs e)
        {

        }
        // -- Button Event Methods
        private void SummaryPageButton_Click(object sender, RoutedEventArgs e)
        {
            CostSummary summary = new CostSummary();

            summary = summaryBroughtForward;

            // -- Run harvest form to get values
            HarvestValues();

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

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

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

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

        }

        // -- Combo SELECTION CHANGED event methods
        private void InsurancePeriodCombo_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            var selectedComboItem = sender as ComboBox;
            insurancePeriod = selectedComboItem.SelectedItem as string;
        }



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



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


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


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

            return myList;
        }

        // -- Harvest data from the form
        private void HarvestValues()
        {
            //-- Insurance
            if (InsuranceCostTextBox.Text != "")
            {
                if (int.TryParse(s: InsuranceCostTextBox.Text, result:
                  out int insuranceValue))
                {
                    insurance = insuranceValue;
                }
                else
                {
                    MessageBox.Show(messageBoxText: "The insurance cost
                  must be a whole number");
                }
            }
            else
            {
                MessageBox.Show(messageBoxText: "Enter the insurance cost,
                  even if it's 0");
            }

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

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

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

        }
    }
}
```