

```
1 using System.Collections.Generic;
2 using System.Windows;
3 using System.Windows.Controls;
4
5 /* TITLE:      PageRunning
6  * AUTHOR:      Paul McKillop
7  * DATE:        October 2022
8  * PURPOSE:     PageRunning interaction code
9  */
10
11 namespace McKillopMotoring
12 {
13     /// <summary>
14     /// Interaction logic for PageRunning.xaml
15     /// </summary>
16     public partial class PageRunning : Page
17     {
18         //-- variables to hold period values selected in the combos
19         //-- Placing them here allows use throughout the module
20         //-- This is a different approach to previous used
21         private string insurancePeriod = "Annual";    //-- Initialised ↗
22         Annual
23         private string fuelPeriod = "Annual";         //-- Ditto
24         private string servicingPeriod = "Annual";    //-- Ditto
25         private string roadTaxPeriod = "Annual";      //-- Ditto
26         //-- Numeric values
27         private int insurance = 0;                    //-- Initialised to 0
28         private int fuel = 0;                         //-- Ditto
29         private int servicing = 0;                    //-- Ditto
30         private int roadTax = 0;                      //-- Ditto
31
32         //-- Combo items list of strings
33         List<string> costPeriods = new List<string>();
34
35         //-- Object to manage data brought forward
36         CostSummary summaryBroughtForward = new CostSummary();
37
38         public PageRunning(CostSummary summaryPassed)
39         {
40             InitializeComponent();
41             // -- Debug message
42             // -- TODO:: Remove message on build
43             // string price =
44             summaryPassed.CurrentLoan.CarPrice.ToString();
45             // MessageBox.Show("The Car Price passed from PageLoan was ↗
46             " + price);
47
48             summaryBroughtForward = summaryPassed;
49             costPeriods = Periods();
50
51             //-- Combo data sources (maybe redundant)
52             InsurancePeriodCombo.ItemsSource = Periods();
53             FuelPeriodCombo.ItemsSource = Periods();
54         }
55     }
56 }
```

```
51      ServicingPeriodCombo.ItemsSource = Periods();
52      RoadTaxPeriodCombo.ItemsSource = Periods();
53
54
55  }
56
57  private void ClearButton_Click(object sender, RoutedEventArgs e)  ➤
58  {
59
60  }
61  // -- Button Event Methods
62  private void SummaryPageButton_Click(object sender,  ➤
63      RoutedEventArgs e)
64  {
65      CostSummary summary = new CostSummary();
66
67      summary = summaryBroughtForward;
68
69      // -- Run harvest form to get values
70      HarvestValues();
71
72      RunningCost runningCost = new RunningCost
73      {
74          Insurance = insurance,
75          InsurancePeriod = insurancePeriod,
76          Fuel = fuel,
77          FuelPeriod = fuelPeriod,
78          Servicing = servicing,
79          ServicingPeriod = servicingPeriod,
80          RoadTax = roadTax,
81          RoadTaxPeriod = roadTaxPeriod
82      };
83
84      summary.RunningCost = runningCost;
85
86      var pageSummary = new PageSummary(summary);
87      this.NavigationService.Navigate(pageSummary);
88  }
89
90  // -- Combo LOAD events methods
91  private void InsurancePeriodCombo_Loaded(object sender,  ➤
92      RoutedEventArgs e)
93  {
94      var combo = sender as ComboBox;
95      combo.ItemsSource = costPeriods;
96      combo.SelectedIndex = 0;
97  }
98
99  private void FuelPeriodCombo_Loaded(object sender,  ➤
100      RoutedEventArgs e)
101  {
102      var combo = sender as ComboBox;
```

```
100         combo.ItemsSource = costPeriods;
101         combo.SelectedIndex = 0;
102     }
103
104     private void ServicingPeriodCombo_Loaded(object sender,      ↗
105         RoutedEventArgs e)
106     {
107         var combo = sender as ComboBox;
108         combo.ItemsSource = costPeriods;
109         combo.SelectedIndex = 0;
110     }
111
112     private void RoadTaxPeriodCombo_Loaded(object sender,      ↗
113         RoutedEventArgs e)
114     {
115         var combo = sender as ComboBox;
116         if (combo != null)
117         {
118             combo.ItemsSource = costPeriods;
119             combo.SelectedIndex = 0;
120         }
121     }
122
123     // -- Combo SELECTION CHANGED event methods
124     private void InsurancePeriodCombo_SelectionChanged(object      ↗
125         sender, SelectionChangedEventArgs e)
126     {
127         var selectedComboItem = sender as ComboBox;
128         insurancePeriod = selectedComboItem.SelectedItem as string;
129     }
130
131     private void FuelPeriodCombo_SelectionChanged(object sender,      ↗
132         SelectionChangedEventArgs e)
133     {
134         var selectedComboItem = sender as ComboBox;
135         fuelPeriod = selectedComboItem.SelectedItem as string;
136     }
137
138
139     private void ServicingPeriodCombo_SelectionChanged(object      ↗
140         sender, SelectionChangedEventArgs e)
141     {
142         var selectedComboItem = sender as ComboBox;
143         servicingPeriod = selectedComboItem.SelectedItem as string;
144     }
145
146     private void RoadTaxPeriodCombo_SelectionChanged(object sender, ↗
147         SelectionChangedEventArgs e)
```

```
147     {
148         var selectedComboItem = sender as ComboBox;
149         roadTaxPeriod = selectedComboItem.SelectedItem as string;
150     }
151
152
153     // -- Utility code
154     // -- Combo box data population values
155     List<string> Periods()
156     {
157         List<string> myList = new List<string>
158         {
159             "Annual",
160             "Monthly",
161             "Weekly"
162         };
163
164         return myList;
165     }
166
167     // -- Harvest data from the form
168     private void HarvestValues()
169     {
170         //-- Insurance
171         if (InsuranceCostTextBox.Text != "")
172         {
173             if (int.TryParse(s: InsuranceCostTextBox.Text, result: out int insuranceValue))
174             {
175                 insurance = insuranceValue;
176             }
177             else
178             {
179                 MessageBox.Show(messageBoxText: "The insurance cost
180                                     must be a whole number");
181             }
182         }
183         else
184         {
185             MessageBox.Show(messageBoxText: "Enter the insurance cost,
186                                     even if it's 0");
187         }
188
189         //-- Fuel
190         if (FuelCostTextBox.Text != "")
191         {
192             if (int.TryParse(s: FuelCostTextBox.Text, result: out int fuelValue))
193             {
194                 fuel = fuelValue;
195             }
196             else
197             {
198                 MessageBox.Show(messageBoxText: "Enter the fuel cost,
199                                     even if it's 0");
200             }
201         }
202     }
203 }
```

```
196         MessageBox.Show(messageBoxText: "The fuel cost must be a whole number");
197     }
198 }
199 else
200 {
201     MessageBox.Show(messageBoxText: "You must enter a fuel cost, even if it's 0");
202 }
203
204 //-- Servicing
205 if (ServicingCostTextBox.Text != "")
206 {
207     if (int.TryParse(s: ServicingCostTextBox.Text, result: out int servicingValue))
208     {
209         servicing = servicingValue;
210     }
211     else
212     {
213         MessageBox.Show(messageBoxText: "The servicing cost must be a whole number, even if it's 0");
214     }
215 }
216 else
217 {
218     MessageBox.Show(messageBoxText: "You must enter a servicing cost, even if it's 0");
219 }
220
221 //-- Road tax
222 if (RoadTaxCostTextBox.Text != "")
223 {
224     if (int.TryParse(s: RoadTaxCostTextBox.Text, result: out int roadTaxValue))
225     {
226         roadTax = roadTaxValue;
227     }
228     else
229     {
230         MessageBox.Show(messageBoxText: "The road tax amount must be a whole number, even if it's 0");
231     }
232 }
233 else
234 {
235     MessageBox.Show(messageBoxText: "you must enter a Road Tax value, even if it's 0");
236 }
237
238 }
239 }
240 }
```