```
1  using System.Windows;
2  using System.Windows.Controls;
3
4  /*  TITLE:      PageLoan
5   *  AUTHOR:     Paul McKillop
6   *  DATE:       October 2022
7   *  PURPOSE:    Page interaction logic
8   */
9
10 namespace McKillopMotoring
11 {
12     /// <summary>
13     /// Interaction logic for PageLoan.xaml
14     /// </summary>
15     public partial class PageLoan : Page
16     {
17         public PageLoan()
18         {
19             InitializeComponent();
20
21             // -- TODO:: Remove at final Build
22             FillTestData();
23         }
24
25         private void RunningPageButton_OnClick(object sender,
           RoutedEventArgs e)
26         {
27             // -- Handling objects for data
28             Loan loan  = new Loan();
29             CostSummary costSummary = new CostSummary();
30
31             // -- Assign Loan through harvest
32             loan = HarvestLoanData();
33             // -- Pass loan to summary
34             costSummary.CurrentLoan = loan;
35
36             // -- Make a page object for navigator
37             var runningCostPage = new PageRunning(summaryPassed:
               costSummary);
38             // -- Navigate to page
39             this.NavigationService.Navigate(root: runningCostPage);
40         }
41
42         private void ClearButton_OnClick(object sender, RoutedEventArgs
              e)
43         {
44             this.CarPriceTextBox.Text = "";
45             this.CarDepositTextBox.Text = "";
46             this.LoanTermTextBox.Text = "";
47             this.InterestRateTextBox.Text = "";
48         }
49
50         // -- Harvest Loan Data
```

```
52          private Loan HarvestLoanData()
53          {
54              Loan loan = new Loan();
55              //-- validate and get data
56
57              if (CarPriceTextBox.Text != "")
58              {
59                  if (int.TryParse(s: CarPriceTextBox.Text, result: out int
                        price))
60                  {
61                      loan.CarPrice = price;
62                  } else
63                  {
64                      //-- Message to the user if the value is not the
                        right type
65                      MessageBox.Show(messageBoxText: "The price entered must
                        be an integer");
66                  }
67              }
68              else
69              {
70                  //-- Message to the user if the TextBox is empty
71                  MessageBox.Show(messageBoxText: "You must provide a price
                        for the car");
72              }
73
74              // -- Deposit
75              if (CarDepositTextBox.Text != "")
76              {
77                  if (int.TryParse(s: CarDepositTextBox.Text, result: out
                        int deposit))
78                  {
79                      loan.CarDeposit = deposit;
80                  }
81                  else
82                  {
83                      MessageBox.Show(messageBoxText: "The deposit value must
                        be an integer");
84                  }
85              }
86              else
87              {
88                  MessageBox.Show(messageBoxText: "Enter an amount for the
                        deposit");
89              }
90
91              // -- Term in years
92              if (LoanTermTextBox.Text != "")
93              {
94                  if (byte.TryParse(s: LoanTermTextBox.Text, result: out
                        byte term))
95                  {
96                      loan.LoanTermYears = term;
```

```
100                     MessageBox.Show(messageBoxText: "The term must be a
                      whole number less than 255");
101                 }
102             }
103             else
104             {
105                 MessageBox.Show(messageBoxText: "You must supply a value
                    for the Loan Term in Years");
106             }
107
108             // -- Interest Rate
109             if (InterestRateTextBox.Text != "")
110             {
111                 if (float.TryParse(s: InterestRateTextBox.Text, result:
                    out float rate))
112                 {
113                     loan.LoanRate = rate;
114                 }
115                 else
116                 {
117                     MessageBox.Show(messageBoxText: "The rate must be
                    formatted as a number");
118                 }
119             }
120             else
121             {
122                 MessageBox.Show(messageBoxText: "You must supply a value
                    for the interest rate");
123             }
124             return loan;
125         }
126
127         private void FillTestData()
128         {
129             Loan myLoan = new Loan
130             {
131                 CarPrice = 15000,
132                 CarDeposit = 2000,
133                 LoanTermYears = 3,
134                 LoanRate = 6.15F
135             };
136
137             this.CarPriceTextBox.Text = myLoan.CarPrice.ToString();
138             this.CarDepositTextBox.Text = myLoan.CarDeposit.ToString();
139             this.LoanTermTextBox.Text = myLoan.LoanTermYears.ToString
                  ();
140             this.InterestRateTextBox.Text = myLoan.LoanRate.ToString
                  (format: "F");
141         }
142     }
143 }
144
```