```
1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4  using System.Windows;
5  using System.Windows.Controls;
6  using System.Windows.Data;
7  using System.Windows.Documents;
8  using System.Windows.Input;
9  using System.Windows.Media;
10 using System.Windows.Media.Imaging;
11 using System.Windows.Navigation;
12 using System.Windows.Shapes;
13 using FlooringCalculator.Models;
14
15 /*
16  * Title:   PageDataEntry
17  * Author:  Paul McKillop
18  * Date:    November 2020
19  * Purpose: Code behind for page
20  */
21
22 /* ***********************
23  * COMPLETION SEQUENCE
24  * ***********************
25  *
26  * This is the most detailed of all the processes.
27  * The order is important because of the dependency of
28  * some methods on others that must already be created.
29  *
30  * Video 24
31  * 00. Check all Gui controls have names
32  * 01. Directive for models
33  * 02. Gui control methods
34  *      a) Clear button
35  *      b) Calculate button
36  *      c) Combo OnLoaded
37  *      d) Combo OnSelectionChanged
38  * 03. Handler data variables
39  * 04. Assignment data for testing
40  * 05. Call assignment data in Page constructor
41  * Video 25
42  * 06. Create a list of tiles for Combo control
43  * 07. Complete combo OnLoaded method
44  * 08. Complete combo OnSelectionChanged method
45  * 09. Create ResetControls method
46  * 10. Call ResetControls from Clear button Click
47  * Video 26
48  * 11. Create method GetSelectedTile
49  * 12. Create method ControlHasValueCheck
50  * 13. Gui Help Button Click method
51  * 14. Implement Help button method
52  * Video 27
53  * 15. HarvestData method
54  * 16. Prepare PageSummary to receive data
55  * Video 28
56  * 17. Implement Calculate Button Click method
57  * 18. Test all
58  */
```

```
59
60  namespace FlooringCalculator
61  {
62      /// <summary>
63      /// Interaction logic for PageDataEntry.xaml
64      /// </summary>
65      public partial class PageDataEntry : Page
66      {
67
68          // -- variables for management of data in the module
69          private string selectedTileName = string.Empty;
70          private Room room = new Room();
71          private Tile tile = new Tile();
72          private DataSummary dataSummary = new DataSummary();
73
74          public PageDataEntry()
75          {
76              InitializeComponent();
77              AssignmentRoomData();
78          }
79
80
81          // -- Clears the form's controls
82          private void ClearButton_OnClick(object sender, RoutedEventArgs e)
83          {
84              ResetControls();
85          }
86
87          private void CalculateButton_OnClick(object sender, RoutedEventArgs e)
88          {
89              // -- Need Calculator object as non static
90              var calculator = new Calculator();
91
92              try
93              {
94                  var controlData = false;
95
96                  controlData = ControlHasValueCheck();
97
98                  if (controlData)
99                  {
100                     // -- Get base data needed for room and tile
101                     tile = GetSelectedTile();
102                     room = HarvestData();
103
104                     // -- create a DataSummary object to hold calculation
                        outcomes
105                     // -- This object will be passed to the PageSummary page
106                     // -- and then displayed in the TextBlock
107
108                     var dataForSummary = new DataSummary()
109                     {
110                         WholeRoomArea = RoomAreas.WholeRoomArea(room).ToString
                        (),
111                         Cutout1Area = RoomAreas.AreaCutout1(room).ToString(),
112                         Cutout2Area = RoomAreas.AreaCutout2(room).ToString(),
113                         TileSizeUsed = selectedTileName,
114                         TilesNeededForRoom = calculator.NumberTilesForFloor
                        (room, tile).ToString(),
115                         LeftoverTileArea = calculator.AreaLeftoverTile(room,
```

```
                    tile).ToString(),
116                     PerimeterLength = RoomAreas.RoomPerimeter
                    (room).ToString()
117                 };
118
119                 dataSummary = dataForSummary;
120
121                 // -- DEBUG Test message
122                 MessageBox.Show(dataSummary.SummaryForDisplay());
123
124                 // -- Navigate to PageSummary with the dataSummary object
125                 var pageSummary = new PageSummary(dataSummary);
126                 var navigationService = NavigationService;
127                 if (navigationService != null) navigationService.Navigate
                    (pageSummary);
128             }
129             else
130             {
131                 MessageBox.Show("Some data is missing. Please check");
132             }
133
134         }
135         catch (Exception exception)
136         {
137             Console.WriteLine(exception);
138             throw;
139         }
140     }
141
142     private void TileComboBox_OnLoaded(object sender, RoutedEventArgs e)
143     {
144         var combo = (ComboBox)sender;
145         if (combo == null) return;
146         combo.ItemsSource = Tiles();
147         combo.SelectedIndex = 0;
148     }
149
150     private void TileComboBox_OnSelectionChanged(object sender,
          SelectionChangedEventArgs e)
151     {
152         var combo = (ComboBox)sender;
153
154         try
155         {
156             if (combo != null) selectedTileName =
                  combo.SelectedItem.ToString();
157         }
158         catch (Exception exception)
159         {
160             Console.WriteLine(exception);
161             throw;
162         }
163     }
164     private void LaunchHelpButton_OnClick(object sender, RoutedEventArgs
          e)
165     {
166         var pageHelp = new PageHelp();
167         if (NavigationService != null) NavigationService.Navigate
              (pageHelp);
168     }
169
```

```
170
171         private void AssignmentRoomData()
172         {
173             RoomWideATextBox.Text = "6.50";
174             RoomLongBTextBox.Text = "7.20";
175             Cutout1WideCTextBox.Text = "1.60";
176             Cutout1LongDTextBox.Text = "2.30";
177             Cutout2WideETextBox.Text = "0.6";
178             Cutout2LongFTextBox.Text = "0.3";
179         }
180
181         /// <summary>
182         /// Make a list of tile types
183         /// </summary>
184         /// <returns></returns>
185         private List<string> Tiles()
186         {
187             List<string> tiles = new List<string>();
188             tiles.Add("60 x 60");
189             tiles.Add("75 x 75");
190
191             return tiles;
192         }
193
194         /// <summary>
195         /// Reset data entry controls
196         /// </summary>
197         private void ResetControls()
198         {
199             RoomWideATextBox.Text = "0";
200             RoomLongBTextBox.Text = "0";
201             Cutout1WideCTextBox.Text = "0";
202             Cutout1LongDTextBox.Text = "0";
203             Cutout2WideETextBox.Text = "0";
204             Cutout2LongFTextBox.Text = "0";
205         }
206
207
208         /// <summary>
209         /// In production, this would search a database for stored
210         /// Tile information
211         /// </summary>
212         /// <returns>Tile object</returns>
213         private Tile GetSelectedTile()
214         {
215             var tempTile = new Tile();
216
217             switch (selectedTileName)
218             {
219                 case "60 x 60":
220                     tempTile.TileWide = 0.60m;
221                     tempTile.TileLong = 0.60m;
222                     break;
223                 case "75 x 75":
224                     tempTile.TileWide = 0.75m;
225                     tempTile.TileLong = .75m;
226                     break;
227                 // -- we must provide a default case allowing for
228                 // -- not in list
```

```
229                 default:
230                     tempTile.TileWide = 1;
231                     tempTile.TileLong = 1;
232                     break;
233             }
234 
235             return tempTile;
236         }
237 
238 
239         /// <summary>
240         /// /// Check that all text box controls have a value in the
241         /// to work with
242         /// </summary>
243         /// <returns>bool true if all have data</returns>
244         private bool ControlHasValueCheck()
245         {
246             return !string.IsNullOrEmpty(RoomWideATextBox.Text) &&
247                    !string.IsNullOrEmpty(RoomLongBTextBox.Text) &&
248                    !string.IsNullOrEmpty(Cutout1WideCTextBox.Text) &&
249                    !string.IsNullOrEmpty(Cutout1LongDTextBox.Text) &&
250                    !string.IsNullOrEmpty(Cutout2WideETextBox.Text) &&
251                    !string.IsNullOrEmpty(Cutout2LongFTextBox.Text);
252         }
253 
254 
255         private Room HarvestData()
256         {
257             try
258             {
259                 // -- very long way round but clearer, perhaps
260                 // -- Get the required values from the Page controls
261                 decimal roomWide = decimal.Parse(RoomWideATextBox.Text);
262                 decimal roomLong = decimal.Parse(RoomLongBTextBox.Text);
263                 decimal cutout1WideC = decimal.Parse
                      (Cutout1WideCTextBox.Text);
264                 decimal cutout1LongD = decimal.Parse
                      (Cutout1LongDTextBox.Text);
265                 decimal cutout2WideE = decimal.Parse
                      (Cutout2WideETextBox.Text);
266                 decimal cutout2LongF = decimal.Parse
                      (Cutout2LongFTextBox.Text);
267 
268                 // -- now initialize an object using the data member
                      parameters
269                 var tempRoom = new Room()
270                 {
271                     RoomWide = roomWide,
272                     RoomLong = roomLong,
273                     Cutout1Wide = cutout1WideC,
274                     Cutout1Long = cutout1LongD,
275                     Cutout2Wide = cutout2WideE,
276                     Cutout2Long = cutout2LongF
277                 };
278 
279                 // -- return the method using the object
280                 return tempRoom;
281             }
282             catch (Exception e)
```

```
283                 {
284                     Console.WriteLine(e);
285                     throw;
286                 }
287             }
288 
289         }
290     }
291 
```