```
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                     else
125                     {
126                         MessageBox.Show("Some data is missing. Please check");
127                     }
128
129                 }
130                 catch (Exception exception)
131                 {
132                     Console.WriteLine(exception);
133                     throw;
134                 }
135             }
136
137             private void TileComboBox_OnLoaded(object sender, RoutedEventArgs e)
138             {
139                 var combo = (ComboBox)sender;
140                 if (combo == null) return;
141                 combo.ItemsSource = Tiles();
142                 combo.SelectedIndex = 0;
143             }
144
145             private void TileComboBox_OnSelectionChanged(object sender,
                  SelectionChangedEventArgs e)
146             {
147                 var combo = (ComboBox)sender;
148
149                 try
150                 {
151                     if (combo != null) selectedTileName =
                          combo.SelectedItem.ToString();
152                 }
153                 catch (Exception exception)
154                 {
155                     Console.WriteLine(exception);
156                     throw;
157                 }
158             }
159             private void LaunchHelpButton_OnClick(object sender, RoutedEventArgs
                  e)
160             {
161                 var pageHelp = new PageHelp();
162                 if (NavigationService != null) NavigationService.Navigate
                      (pageHelp);
163             }
164
165
166             private void AssignmentRoomData()
167             {
168                 RoomWideATextBox.Text = "6.50";
169                 RoomLongBTextBox.Text = "7.20";
170                 Cutout1WideCTextBox.Text = "1.60";
```

```
171             Cutout1LongDTextBox.Text = "2.30";
172             Cutout2WideETextBox.Text = "0.6";
173             Cutout2LongFTextBox.Text = "0.3";
174         }
175
176         /// <summary>
177         /// Make a list of tile types
178         /// </summary>
179         /// <returns></returns>
180         private List<string> Tiles()
181         {
182             List<string> tiles = new List<string>();
183             tiles.Add("60 x 60");
184             tiles.Add("75 x 75");
185
186             return tiles;
187         }
188
189         /// <summary>
190         /// Reset data entry controls
191         /// </summary>
192         private void ResetControls()
193         {
194             RoomWideATextBox.Text = "0";
195             RoomLongBTextBox.Text = "0";
196             Cutout1WideCTextBox.Text = "0";
197             Cutout1LongDTextBox.Text = "0";
198             Cutout2WideETextBox.Text = "0";
199             Cutout2LongFTextBox.Text = "0";
200         }
201
202
203         /// <summary>
204         /// In production, this would search a database for stored
205         /// Tile information
206         /// </summary>
207         /// <returns>Tile object</returns>
208         private Tile GetSelectedTile()
209         {
210             var tempTile = new Tile();
211
212             switch (selectedTileName)
213             {
214                 case "60 x 60":
215                     tempTile.TileWide = 0.60m;
216                     tempTile.TileLong = 0.60m;
217                     break;
218                 case "75 x 75":
219                     tempTile.TileWide = 0.75m;
220                     tempTile.TileLong = .75m;
221                     break;
222                 // -- we must provide a default case allowing for
223                 // -- not in list
224                 default:
225                     tempTile.TileWide = 1;
226                     tempTile.TileLong = 1;
227                     break;
228             }
229
```

```
230             return tempTile;
231         }
232
233
234         /// <summary>
235         /// /// Check that all text box controls have a value in the
236         /// to work with
237         /// </summary>
238         /// <returns>bool true if all have data</returns>
239         private bool ControlHasValueCheck()
240         {
241             return !string.IsNullOrEmpty(RoomWideATextBox.Text) &&
242                    !string.IsNullOrEmpty(RoomLongBTextBox.Text) &&
243                    !string.IsNullOrEmpty(Cutout1WideCTextBox.Text) &&
244                    !string.IsNullOrEmpty(Cutout1LongDTextBox.Text) &&
245                    !string.IsNullOrEmpty(Cutout2WideETextBox.Text) &&
246                    !string.IsNullOrEmpty(Cutout2LongFTextBox.Text);
247         }
248
249
250         private Room HarvestData()
251         {
252             try
253             {
254                 // -- very long way round but clearer, perhaps
255                 // -- Get the required values from the Page controls
256                 decimal roomWide = decimal.Parse(RoomWideATextBox.Text);
257                 decimal roomLong = decimal.Parse(RoomLongBTextBox.Text);
258                 decimal cutout1WideC = decimal.Parse
                      (Cutout1WideCTextBox.Text);
259                 decimal cutout1LongD = decimal.Parse
                      (Cutout1LongDTextBox.Text);
260                 decimal cutout2WideE = decimal.Parse
                      (Cutout2WideETextBox.Text);
261                 decimal cutout2LongF = decimal.Parse
                      (Cutout2LongFTextBox.Text);
262
263                 // -- now initialize an object using the data member
                      parameters
264                 var tempRoom = new Room()
265                 {
266                     RoomWide = roomWide,
267                     RoomLong = roomLong,
268                     Cutout1Wide = cutout1WideC,
269                     Cutout1Long = cutout1LongD,
270                     Cutout2Wide = cutout2WideE,
271                     Cutout2Long = cutout2LongF
272                 };
273
274                 // -- return the method using the object
275                 return tempRoom;
276             }
277             catch (Exception e)
278             {
279                 Console.WriteLine(e);
280                 throw;
281             }
282         }
283
284     }
```

```
285  }
286
```