```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using FlooringCalculator.Models;

/*
 * Title:   PageDataEntry
 * Author:  Paul McKillop
 * Date:    November 2020
 * Purpose: Code behind for page
 */

/* ***********************
 * COMPLETION SEQUENCE
 * ***********************
 *
 * This is the most detailed of all the processes.
 * The order is important because of the dependency of
 * some methods on others that must already be created.
 *
 * Video 24
 * 00. Check all Gui controls have names
 * 01. Directive for models
 * 02. Gui control methods
 *      a) Clear button
 *      b) Calculate button
 *      c) Combo OnLoaded
 *      d) Combo OnSelectionChanged
 * 03. Handler data variables
 * 04. Assignment data for testing
 * 05. Call assignment data in Page constructor
 * Video 25
 * 06. Create a list of tiles for Combo control
 * 07. Complete combo OnLoaded method
 * 08. Complete combo OnSelectionChanged method
 * 09. Create ResetControls method
 * 10. Call ResetControls from Clear button Click
 * Video 26
 * 11. Create method GetSelectedTile
 * 12. Create method ControlHasValueCheck
 * 13. Gui Help Button Click method
 * 14. Implement Help button method
 * Video 27
 * 15. HarvestData method
 * 16. Prepare PageSummary to receive data
 * Video 28
 * 17. Implement Calculate Button Click method
 * 18. Test all
 */
```

```

namespace FlooringCalculator
{
    /// <summary>
    /// Interaction logic for PageDataEntry.xaml
    /// </summary>
    public partial class PageDataEntry : Page
    {

        // -- variables for management of data in the module
        private string selectedTileName = string.Empty;
        private Room room = new Room();
        private Tile tile = new Tile();
        private DataSummary dataSummary = new DataSummary();

        public PageDataEntry()
        {
            InitializeComponent();
            AssignmentRoomData();
        }


        // -- Clears the form's controls
        private void ClearButton_OnClick(object sender, RoutedEventArgs e)
        {
            ResetControls();
        }

        private void CalculateButton_OnClick(object sender, RoutedEventArgs e)
        {

        }

        private void TileComboBox_OnLoaded(object sender, RoutedEventArgs e)
        {
            var combo = (ComboBox)sender;
            if (combo == null) return;
            combo.ItemsSource = Tiles();
            combo.SelectedIndex = 0;
        }

        private void TileComboBox_OnSelectionChanged(object sender,
          SelectionChangedEventArgs e)
        {
            var combo = (ComboBox)sender;

            try
            {
                if (combo != null) selectedTileName =
                  combo.SelectedItem.ToString();
            }
            catch (Exception exception)
            {
                Console.WriteLine(exception);
                throw;
            }
        }
        private void LaunchHelpButton_OnClick(object sender, RoutedEventArgs
          e)
        {
            var pageHelp = new PageHelp();
```

```
                if (NavigationService != null) NavigationService.Navigate
                  (pageHelp);
            }


            private void AssignmentRoomData()
            {
                RoomWideATextBox.Text = "6.50";
                RoomLongBTextBox.Text = "7.20";
                Cutout1WideCTextBox.Text = "1.60";
                Cutout1LongDTextBox.Text = "2.30";
                Cutout2WideETextBox.Text = "0.6";
                Cutout2LongFTextBox.Text = "0.3";
            }

            /// <summary>
            /// Make a list of tile types
            /// </summary>
            /// <returns></returns>
            private List<string> Tiles()
            {
                List<string> tiles = new List<string>();
                tiles.Add("60 x 60");
                tiles.Add("75 x 75");

                return tiles;
            }

            /// <summary>
            /// Reset data entry controls
            /// </summary>
            private void ResetControls()
            {
                RoomWideATextBox.Text = "0";
                RoomLongBTextBox.Text = "0";
                Cutout1WideCTextBox.Text = "0";
                Cutout1LongDTextBox.Text = "0";
                Cutout2WideETextBox.Text = "0";
                Cutout2LongFTextBox.Text = "0";
            }


            /// <summary>
            /// In production, this would search a database for stored
            /// Tile information
            /// </summary>
            /// <returns>Tile object</returns>
            private Tile GetSelectedTile()
            {
                var tempTile = new Tile();

                switch (selectedTileName)
                {
                    case "60 x 60":
                        tempTile.TileWide = 0.60m;
                        tempTile.TileLong = 0.60m;
                        break;
                    case "75 x 75":
                        tempTile.TileWide = 0.75m;
```

```csharp
                    tempTile.TileLong = .75m;
                    break;
                // -- we must provide a default case allowing for
                // -- not in list
                default:
                    tempTile.TileWide = 1;
                    tempTile.TileLong = 1;
                    break;
            }

            return tempTile;
        }


        /// <summary>
        /// /// Check that all text box controls have a value in the
        /// to work with
        /// </summary>
        /// <returns>bool true if all have data</returns>
        private bool ControlHasValueCheck()
        {
            return !string.IsNullOrEmpty(RoomWideATextBox.Text) &&
                   !string.IsNullOrEmpty(RoomLongBTextBox.Text) &&
                   !string.IsNullOrEmpty(Cutout1WideCTextBox.Text) &&
                   !string.IsNullOrEmpty(Cutout1LongDTextBox.Text) &&
                   !string.IsNullOrEmpty(Cutout2WideETextBox.Text) &&
                   !string.IsNullOrEmpty(Cutout2LongFTextBox.Text);
        }


        private Room HarvestData()
        {
            try
            {
                // -- very long way round but clearer, perhaps
                // -- Get the required values from the Page controls
                decimal roomWide = decimal.Parse(RoomWideATextBox.Text);
                decimal roomLong = decimal.Parse(RoomLongBTextBox.Text);
                decimal cutout1WideC = decimal.Parse(Cutout1WideCTextBox.Text);
                decimal cutout1LongD = decimal.Parse(Cutout1LongDTextBox.Text);
                decimal cutout2WideE = decimal.Parse(Cutout2WideETextBox.Text);
                decimal cutout2LongF = decimal.Parse(Cutout2LongFTextBox.Text);

                // -- now initialize an object using the data member parameters
                var tempRoom = new Room()
                {
                    RoomWide = roomWide,
                    RoomLong = roomLong,
                    Cutout1Wide = cutout1WideC,
                    Cutout1Long = cutout1LongD,
                    Cutout2Wide = cutout2WideE,
                    Cutout2Long = cutout2LongF
                };

```

```
                    // -- return the method using the object
                    return tempRoom;
                }
                catch (Exception e)
                {
                    Console.WriteLine(e);
                    throw;
                }
            }

        }
    }

```