GUI Application Exercises

1) PassengerDetails Add Window

Add a new WPF Application to the solution and give it the name FlightWpf.

To the project add a new Window (WPF) and call it AddWindow with title ‘Add Passenger’. Set the WindowStartupLocation property to CenterOwner. Set the ResizeMode property to NoResize. To this new window add two text boxes and two labels to allow entering of name and weight (passenger details). Change the names of the text boxes appropriately.

Add radio buttons to allow selection of “Ordinary” or “Silver” passenger details. Default the “Ordinary” option to being checked. Change the names of the radio buttons appropriately. Add buttons for OK and Cancel to the window. Set the IsCancel property of the Cancel button to true. Within the XAML bind the Text properties to Name and Weight properties.

2) Display Add Window Exercise

Add a button to the MainWindow and in the Click event handler add the code:

C#:

```
AddWindow win = new AddWindow();

if( win.ShowDialog() ?? false)
{ ... }
```

VB.NET:

```
Dim win As New AddWindow()

if win.ShowDialog().GetValueOrDefault( False) Then
End If
```

To the MainWindow add a PassengerDetails field and instantiate a PassengerDetails. Within the Click event handler assign the PassengerDetails to the AddWindow DataContext property.

Build and test.

3) Create Builder Class Exercise

Add a new public class called PassengerDetailsBuilder to the FlightWpf project. Create public enum called PassengerType with values ‘Ordinary’ and ‘Silver’.

To the PassengerDetailsBuilder class add properties for Name (string), Weight (int). Also add properties for IsOrdinary and IsSilver to set the PassengerType field. Add a method called GetPassengerDetails(). Within the GetPassengerDetails() method instantiate PassengerDetails or SilverPassengerDetails dependent upon the passenger type.

Edit the AddPassenger window XAML. For the AddPassenger window bind the radio buttons IsChecked property to the IsOrdinary or IsSilver properties of a PassengerDetailsBuilder.

4) Wire-up Builder Exercise

Within the MainWindow add a PassengerDetailsBuilder field and instantiate a PassengerDetailsBuilder.

Within the MainWindow button Click event handler assign the Builder to the AddWindow DataContext property.

Builder and test.

5) MainWindow Display Exercise

Add a Menu to the top of the Grid within the MainWindow. Edit the Items collection of the Menu to add MenuItems Edit and Help. Edit the Items collection of Edit option to add Add, Delete, ‘Separator’ and Exit options. Edit the Items collection of Help to add an About option.

Add a ListBox to the Grid within the MainWindow.

Build and test.

6) Create ViewModel Exercise

Create a class called PassengerViewModel. Add a Property called Passengers with a backing field for an ObservableCollection of PassengerDetails.

Create a class for a command implementing ICommand interface called AddPassenger. Move the code for displaying the AddWindow window to the Execute method of the command. Within the CanExecute method return ‘true’.

7) ListBox Display Exercise

To the ListBox XAML add a ListBoxItem template element. To this add a DataTemplate element. To this DataTemplate add a StackPanel and within this add two TextBlock elements. Bind the TextBlock elements to Name and Weight properties.

Bind the AddPassenger command to the Add menu item. Associate the Passengers property of a PassengerViewModel with ItemsSource property of the ListBox.

Build and test.