Yung Nguyen

CECS 475 Lab 4

Transfer data update view -> main view

ViewModelLocater.cs

```
public class ViewModelLocator
	{
	/// <summary>
	/// Initializes a new instance of the ViewModelLocator class.
	/// </summary>
	public ViewModelLocator()
	{
		ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);
		SimpleIoc.Default.Register<MainViewModel>();
		Messenger.Default.Register<NotificationMessage>(this, NotifyUserMethod);
	}

	/// <summary>
	/// A property that lets the main window connect with its View Model.
	/// </summary>
	public MainViewModel Main
	{
		get
		{
		return ServiceLocator.Current.GetInstance<MainViewModel>();
		}
	}
	private void NotifyUserMethod(NotificationMessage message)
	{
		if (message.Notification != "Delete") {
MessageBox.Show(message.Notification); }
	}


	}
}
```

ChangeWindow.xaml

```
<Window x:Class="Lab_4_Problem_2.View.ChangeWindow"
	Name="changeWindow"
...
TextWrapping="Wrap" VerticalAlignment="Center" Text="{Binding EnteredFName}"
Width="120" Grid.Row="1" Grid.Column="1"/>
	<TextBox x:Name="textBox1" HorizontalAlignment="Center" Height="23"
TextWrapping="Wrap" VerticalAlignment="Center" Text="{Binding EnteredLName}"
Width="120" Grid.Row="2" Grid.Column="1"/>
	<TextBox x:Name="textBox2" HorizontalAlignment="Center" Height="23"
TextWrapping="Wrap" VerticalAlignment="Center" Text="{Binding EnteredEmail}"
Width="120" Grid.Row="3" Grid.Column="1"/>
```

```
      <Button x:Name="button" Content="Update" HorizontalAlignment="Center"
VerticalAlignment="Top" Command="{Binding UpdateCommand}" CommandParameter="{Binding
ElementName=changeWindow}" Width="75" Grid.Row="4" Grid.Column="0"/>
      <Button x:Name="button1" Content="Delete" HorizontalAlignment="Center"
VerticalAlignment="Top" Command="{Binding DeleteCommand}" CommandParameter="{Binding
ElementName=changeWindow}" Width="75" Grid.Row="4" Grid.Column="1"/>

      </Grid>
</Window>
```

ChangeViewModel.cs

```
…
 public ChangeViewModel()
      {
            UpdateCommand = new RelayCommand<IClosable>(UpdateMethod);
            DeleteCommand = new RelayCommand<IClosable>(DeleteMethod);
            //_____________________________________
            Messenger.Default.Register<Member>(this, GetSelected);
      }
public void UpdateMethod(IClosable window)
      {
            try
            {
            Messenger.Default.Send(new
MessageMember(enteredFName,enteredLName,enteredEmail,"Update"));
            Messenger.Default.Send<NotificationMessage>(new
NotificationMessage("Employees updated!."));
            window.Close();
            }
            catch (ArgumentException)
            {
            MessageBox.Show("Fields must be under 25 characters.", "Entry Error");
            }
            catch (NullReferenceException)
            {
            MessageBox.Show("Fields cannot be empty.", "Entry Error");
            }
            catch (FormatException)
            {
            MessageBox.Show("Must be a valid e-mail address.", "Entry Error");
            }
      }
...
```

MainViewModel.cs

```
…
 public void ReceiveMember(MessageMember m)
      {
            if (m.Message == "Update")
```

```
                {
                var index = MemberList.IndexOf(SelectedMember);

                if (index != -1)
                        MemberList[index] = m;
                        selectedMember = m;
                        this.RaisePropertyChanged(() => this.MemberList);
                        database.SaveMemberships();
                }
                else if (m.Message == "Add")
                {
                members.Add(m);
                this.RaisePropertyChanged(() => this.MemberList);
                database.SaveMemberships();
                }
        }
```

MemberDB.cs (update the text file)

```
…
public void SaveMemberships()
        {
                StreamWriter output = new StreamWriter(new FileStream(filepath,
FileMode.Create, FileAccess.Write));
                foreach (var member in members)
                {
                output.WriteLine(member.FirstName + " " + member.LastName + " " +
member.Email);
                }
                output.Close();
        }
…
```

Transfer data from main view -> update view

```
	<ListBox x:Name="listBox" ItemsSource="{Binding MemberList}"
SelectedItem="{Binding SelectedMember}" Grid.Row ="1" Grid.Column ="0"
HorizontalAlignment="Center" VerticalAlignment="Center" Height="220" Width="322">
		<i:Interaction.Triggers>
		<i:EventTrigger EventName="MouseDoubleClick">
			<i:InvokeCommandAction Command="{Binding ChangeCommand}"/>
		</i:EventTrigger>
		</i:Interaction.Triggers>
	</ListBox>
	<Label x:Name="label" Content="Customers:" Grid.Row ="0" Grid.Column ="0"
HorizontalAlignment="Left" VerticalAlignment="Bottom"/>

	<Grid.ColumnDefinitions>
		<ColumnDefinition Width="380"/>
		<ColumnDefinition />
	</Grid.ColumnDefinitions>

	<Grid.RowDefinitions>
		<RowDefinition Height="*"/>
		<RowDefinition Height="250"/>
		<RowDefinition Height="*"/>
	</Grid.RowDefinitions>

	</Grid>
</Window>
```

MainViewModel.cs

```
…
public MainViewModel()
	{
		members = new ObservableCollection<Member>();
		database = new MemberDB(members);
		members = database.GetMemberships(); //get information from text file
		AddCommand = new RelayCommand(AddMethod);
		ExitCommand = new RelayCommand<IClosable>(ExitMethod);
		ChangeCommand = new RelayCommand(ChangeMethod);
		Messenger.Default.Register<MessageMember>(this, ReceiveMember);
		Messenger.Default.Register<NotificationMessage>(this, ReceiveMessage);
	}
//the selected member
public Member SelectedMember
	{
		get
		{
		return selectedMember;
		}
		set
```

```
                {
                selectedMember = value;
                RaisePropertyChanged("SelectedMember");
                }
        }
//open the change window
public void ChangeMethod()
        {
                if (SelectedMember != null)
                {
                ChangeWindow change = new ChangeWindow();
                change.Show();
                Messenger.Default.Send(SelectedMember);
                }
        }
…
```