Lab Assignment 5
Jerry Belmonte
Keira Wong

CODE (x2 files)

MODEL ******************************

Member.cs

```
using GalaSoft.MvvmLight;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GymMembers.Model
{
    /// <summary>
    /// A class that represents a member of a gym.
    /// </summary>
    public class Member : ObservableObject
    {
        private const int TEXT_LIMIT = 25;

        /// <summary>
        /// The member's first name.
        /// </summary>
        private string firstName;
        /// <summary>
        /// The member's last name.
        /// </summary>
        private string lastName;
        /// <summary>
        /// The member's e-mail.
        /// </summary>
        private string email;

        public Member() { }
```

```
/// <summary>
/// Creates a new member.
/// </summary>
/// <param name="fName">The member's first name.</param>
/// <param name="lName">The member's last name.</param>
/// <param name="mail">The member's e-mail.</param>
public Member(string fName, string lName, string mail)
{
    FirstName = fName;
    LastName = lName;
    Email = mail;
}

/// <summary>
/// A property that gets or sets the member's first name, and makes sure it's not too long.
/// </summary>
public string FirstName
{
    get { return firstName; }
    set
    {
        if (value.Length > TEXT_LIMIT)
        {
            throw new ArgumentException("Too long");
        }

        if (value.Length == 0)
        {
            throw new NullReferenceException();
        }

        firstName = value;
    }
}

/// <summary>
/// A property that gets or sets the member's last name, and makes sure it's not too long.
/// </summary>
public string LastName
{
    get { return lastName; }
    set
    {
        if (value.Length > TEXT_LIMIT)
        {
            throw new ArgumentException("Too long");
        }

        if (value.Length == 0)
        {
            throw new NullReferenceException();
        }

        lastName = value;
    }
}
```

```
        /// <summary>
        /// A property that gets or sets the member's e-mail, and makes sure it's not too long.
        /// </summary>
        public string Email
        {
            get { return email; }
            set
            {
                if (value.Length > TEXT_LIMIT)
                {
                    throw new ArgumentException("Too long");
                }

                if (value.Length == 0)
                {
                    throw new NullReferenceException();
                }

                if (value.IndexOf("@") == -1 || value.IndexOf(".") == -1)
                {
                    throw new FormatException();
                }

                email = value;
            }
        }

        /// <summary>
        /// Text to be displayed in the list box.
        /// </summary>
        /// <returns>A concatenation of the member's first name, last name, and e-mail.</returns>
        public override string ToString() => $"{FirstName} {LastName}, {Email}";
    }
}
```

MemberDB.cs

```
using GalaSoft.MvvmLight;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GymMembers.Model
{
    /// <summary>
    /// A class that uses a text file to store information about the gym members long-term.
    /// </summary>
```

```
class MemberDB : ObservableObject
{
    /// <summary>
    /// The list of members to be saved.
    /// </summary>
    private ObservableCollection<Member> members;

    /// <summary>
    /// Where the database is stored.
    /// </summary>
    private const string filepath = "../members.txt";

    /// <summary>
    /// Creates a new member database.
    /// </summary>
    /// <param name="m">The list to saved from or written to.</param>
    public MemberDB(ObservableCollection<Member> m)
    {
        members = m;
    }

    /// <summary>
    /// Reads the saved text file database into the program's list of members.
    /// </summary>
    /// <returns>The list containing the text file data read in.</returns>
    public ObservableCollection<Member> GetMemberships()
    {
        try
        {
            char[] separators = new char[] { ' ', ',' };
            var fileData = from line in File.ReadLines(filepath)
                           let parts = line.Split(separators, StringSplitOptions.RemoveEmptyEntries)
                           select new Member(parts[0], parts[1], parts[2]);

            members = new ObservableCollection<Member>(fileData);
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine("File not found");
        }
        catch (FormatException)
        {
            Console.WriteLine("Invalid e-mail address format.");
        }
        return members;
    }

    /// <summary>
    /// Saves the program's list of members into the text file database.
    /// </summary>
    public void SaveMemberships()
    {
        File.WriteAllLines(filepath, EnumerateMembers());
    }

    private IEnumerable<string> EnumerateMembers() => from m in members select m.ToString();
    }
}
```

MessageMember.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GymMembers.Model
{
    /// <summary>
    /// An extension of member that also includes a message for some sort of extra description.
    /// </summary>
    public class MessageMember : Member
    {
        /// <summary>
        /// Creates a new member.
        /// </summary>
        /// <param name="fName">The member's first name.</param>
        /// <param name="lName">The member's last name.</param>
        /// <param name="mail">The member's e-mail.</param>
        /// <param name="message">The extra description</param>
        public MessageMember(string fName, string lName, string mail, string message) : base(fName,
lName, mail)
        {
            Message = message;
        }

        /// <summary>
        /// A property that includes the message.
        /// </summary>
        public string Message { get; private set; }
    }
}
```

VIEW **************************

AddWindow.xaml

```
<Window x:Class="GymMembers.View.AddWindow"
        Name="addWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:GymMembers.View"
        mc:Ignorable="d"
        DataContext="{Binding Source={StaticResource Locator}, Path=AddViewModel}"
        Title="Add Membership" Height="237" Width="300">
    <Grid Background="#FFECEAEA">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="10"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="30"/>
        </Grid.RowDefinitions>
        <!-- Labels -->
        <Label x:Name="label1" Content="First Name:" HorizontalAlignment="Center"
VerticalAlignment="Center" Grid.Row="1" Grid.Column="0"/>
        <Label x:Name="label2" Content="Last Name:" HorizontalAlignment="Center"
VerticalAlignment="Center" Grid.Row="2" Grid.Column="0"/>
        <Label x:Name="label3" Content="E-mail:" HorizontalAlignment="Center"
VerticalAlignment="Center" Grid.Row="3" Grid.Column="0"/>
        <!-- TextBoxes -->
        <TextBox x:Name="textBox1" Text="{Binding EnteredFName}" TextWrapping="Wrap"
HorizontalAlignment="Center" VerticalAlignment="Center" Height="23" Width="120" Grid.Row="1"
Grid.Column="1"/>
        <TextBox x:Name="textBox2" Text="{Binding EnteredLName}" TextWrapping="Wrap"
HorizontalAlignment="Center" VerticalAlignment="Center" Height="23" Width="120" Grid.Row="2"
Grid.Column="1"/>
        <TextBox x:Name="textBox3" Text="{Binding EnteredEmail}" TextWrapping="Wrap"
HorizontalAlignment="Center" VerticalAlignment="Center" Height="23" Width="120" Grid.Row="3"
Grid.Column="1"/>
        <!-- Buttons -->
        <Button x:Name="button1" Content="Save" Command="{Binding SaveCommand}"
CommandParameter="{Binding ElementName=addWindow}" HorizontalAlignment="Center"
VerticalAlignment="Top" Width="75" Grid.Row="4" Grid.Column="0"/>
        <Button x:Name="button2" Content="Cancel" Command="{Binding CancelCommand}"
CommandParameter="{Binding ElementName=addWindow}" HorizontalAlignment="Center"
VerticalAlignment="Top" Width="75" Grid.Row="4" Grid.Column="1"/>
    </Grid>
</Window>
```

ChangeWindow.xaml

```
<Window x:Class="GymMembers.View.ChangeWindow"
        Name="changeWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:GymMembers.View"
        mc:Ignorable="d"
        DataContext="{Binding Source={StaticResource Locator}, Path=ChangeViewModel}"
        Title="Change Membership" Height="237" Width="300">
    <Grid Background="#FFECEAEA">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="10"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="30"/>
        </Grid.RowDefinitions>
        <!-- Labels -->
        <Label x:Name="label1" Content="First Name:" HorizontalAlignment="Center"
VerticalAlignment="Center" Grid.Row="1" Grid.Column="0"/>
        <Label x:Name="label2" Content="Last Name:" HorizontalAlignment="Center"
VerticalAlignment="Center" Grid.Row="2" Grid.Column="0"/>
        <Label x:Name="label3" Content="E-mail:" HorizontalAlignment="Center"
VerticalAlignment="Center" Grid.Row="3" Grid.Column="0"/>
        <!-- TextBoxes -->
        <TextBox x:Name="textBox1" Text="{Binding EnteredFName}" TextWrapping="Wrap"
HorizontalAlignment="Center" VerticalAlignment="Center" Height="23" Width="120" Grid.Row="1"
Grid.Column="1"/>
        <TextBox x:Name="textBox2" Text="{Binding EnteredLName}" TextWrapping="Wrap"
HorizontalAlignment="Center" VerticalAlignment="Center" Height="23" Width="120" Grid.Row="2"
Grid.Column="1"/>
        <TextBox x:Name="textBox3" Text="{Binding EnteredEmail}" TextWrapping="Wrap"
HorizontalAlignment="Center" VerticalAlignment="Center" Height="23" Width="120" Grid.Row="3"
Grid.Column="1"/>
        <!-- Buttons -->
        <Button x:Name="button1" Content="Update" HorizontalAlignment="Center" VerticalAlignment="Top"
Command="{Binding UpdateCommand}" CommandParameter="{Binding
ElementName=changeWindow}" Width="75" Grid.Row="4" Grid.Column="0"/>
        <Button x:Name="button2" Content="Delete" HorizontalAlignment="Center" VerticalAlignment="Top"
Command="{Binding DeleteCommand}" CommandParameter="{Binding ElementName=changeWindow}"
Width="75" Grid.Row="4" Grid.Column="1"/>
    </Grid>
</Window>
```

MainWindow.xaml

```
<Window x:Class="GymMembers.View.MainWindow"
        Name="mainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:vm="clr-namespace:GymMembers.ViewModel"
        xmlns:local="clr-namespace:GymMembers"
        xmlns:i="clr-namespace:System.Windows.Interactivity;assembly=System.Windows.Interactivity"
        xmlns:scm1="clr-namespace:System.ComponentModel;assembly=WindowsBase"
        mc:Ignorable="d"
        DataContext="{Binding Source={StaticResource Locator}, Path=MainViewModel}"
        Title="Gym Memberships" Height="350" Width="525">
    <Grid Background="#FFECEAEA">
        <Grid.Resources>
            <CollectionViewSource x:Key="SortedItems" Source="{Binding MemberList}" >
                <CollectionViewSource.SortDescriptions>
                    <scm1:SortDescription PropertyName="FirstName"/>
                </CollectionViewSource.SortDescriptions>
            </CollectionViewSource>
        </Grid.Resources>

        <Button x:Name="button1" Content="Add"
                Grid.Row="1" Grid.Column="1"
                Command="{Binding AddCommand}"
                HorizontalAlignment="Left" VerticalAlignment="Top"
                Width="75" Margin="0,50,0,0"/>

        <Button x:Name="button2" Content="Exit"
                Grid.Row="1" Grid.Column="1"
                Command="{Binding ExitCommand, Mode=OneWay}"
                CommandParameter="{Binding ElementName=mainWindow}"
                HorizontalAlignment="Left" VerticalAlignment="Bottom"
                Width="75" Margin="0,0,0,50"/>

        <ListBox x:Name="listBox"
                 Grid.Row="1" Grid.Column="0"
                 ItemsSource="{Binding Source={StaticResource SortedItems}}"
                 SelectedItem="{Binding SelectedMember}"
                 HorizontalAlignment="Center" VerticalAlignment="Center"
                 Height="220" Width="322">

            <i:Interaction.Triggers>
                <i:EventTrigger EventName="MouseUp">
                    <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>
```

VIEWMODEL ******************

AddViewModel.cs

```
using GalaSoft.MvvmLight;
using GalaSoft.MvvmLight.Command;
using GalaSoft.MvvmLight.Messaging;
using GymMembers.Model;
using System;
using System.Collections.ObjectModel;
using System.IO;
using System.Windows;
using System.Windows.Input;

// KEIRA: (ErrorWindow.xaml Pop-Up) Add namespaces.
using GymMembers.View;
using Prism.Mvvm;
using Prism.Commands;
using System.Windows.Data;

namespace GymMembers.ViewModel
{
    /// <summary>
    /// The VM for adding users to the list.
    /// </summary>
    public class AddViewModel : ViewModelBase
    {
        /// <summary>
        /// The currently entered first name in the add window.
        /// </summary>
        private string enteredFName;

        /// <summary>
        /// The currently entered last name in the add window.
        /// </summary>
        private string enteredLName;

        /// <summary>
        /// The currently entered email in the add window.
        /// </summary>
        private string enteredEmail;

        /// <summary>
        /// Initializes a new instance of the AddViewModel class.
        /// </summary>
        public AddViewModel()
        {
            SaveCommand = new RelayCommand<IClosable>(SaveMethod);
            // KEIRA: (CancelCommand) Attach CancelCommand to CancelMethod to act as an event.
            CancelCommand = new RelayCommand<IClosable>(CancelMethod);
        }

        /// <summary>
        /// The command that triggers saving the filled out member data.
        /// </summary>
        public ICommand SaveCommand { get; private set; }

        /// <summary>
        /// The command that triggers closing the add window.
        /// </summary>
        ///
        // KEIRA: (CancelCommand) Add CancelCommand.
        public ICommand CancelCommand { get; private set; } // REVIEW: Equivalent to "public
    RelayCommand<IClosable> ExitCommand { get; private set; }" ?

        /// <summary>
        /// Sends a valid member to the Main VM to add to the list, then closes the window.
        /// </summary>
        /// <param name="window">The window to close.</param>
        public void SaveMethod(IClosable window)
        {
            try
            {
                if (window != null)
                {
                    var addViewModelMessage = new MessageMember(EnteredFName, EnteredLName,
    EnteredEmail, "Add");
                    Messenger.Default.Send(addViewModelMessage); // sends "Add" message to
    MainViewModel.ReceiveMember(MessageMember m)
                    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");
            }
        }

        /// <summary>
        /// Closes the window.
        /// </summary>
        /// <param name="window">The window to close.</param>
        ///
        // KEIRA: (CancelCommand) Add CancelMethod().
        public void CancelMethod(IClosable window)
        {
            if (window != null)
            {
                window.Close();
            }
        }

        /// <summary>
        /// The currently entered first name in the add window.
        /// </summary>
        public string EnteredFName
        {
            get { return enteredFName; }
            set
            {
                enteredFName = value;
                RaisePropertyChanged("EnteredFName");
            }
        }

        /// <summary>
        /// The currently entered last name in the add window.
        /// </summary>
        public string EnteredLName
        {
            get { return enteredLName; }
            set
            {
                enteredLName = value;
                RaisePropertyChanged("EnteredLName");
            }
        }

        /// <summary>
        /// The currently entered e-mail in the add window.
        /// </summary>
        public string EnteredEmail
        {
            get { return enteredEmail; }
            set
            {
                enteredEmail = value;
                RaisePropertyChanged("EnteredEmail");
            }
        }
    }
}
```

ChangeView.xaml

```
using GalaSoft.MvvmLight;
using GalaSoft.MvvmLight.Command;
using GalaSoft.MvvmLight.Messaging;
using GymMembers.Model;
using System;
using System.Collections.ObjectModel;
using System.IO;
using System.Windows;
using System.Windows.Input;

// REVIEW: (UpdateCommand) Add namespaces.
using GymMembers.View;
using Prism.Mvvm;
using Prism.Commands;
using System.Windows.Data;

namespace GymMembers.ViewModel
{
    /// <summary>
    /// The VM for modifying or removing users.
    /// </summary>
    public class ChangeViewModel : ViewModelBase
    {
        /// <summary>
        /// The currently entered first name in the change window.
        /// </summary>
        private string enteredFName;

        /// <summary>
        /// The currently entered last name in the change window.
        /// </summary>
        private string enteredLName;

        /// <summary>
        /// The currently entered email in the change window.
        /// </summary>
        private string enteredEmail;

        /// <summary>
        /// Initializes a new instance of the ChangeViewModel class.
        /// </summary>
        public ChangeViewModel()
        {
            //GetSelected();
            // KEIRA: (UpdateCommand) Attach UpdateCommand to UpdateMethod to act as an event.
            UpdateCommand = new RelayCommand<IClosable>(UpdateMethod);
            // KEIRA: (DeleteCommand) Attach DeleteCommand to DeleteMethod to act as an event.
            DeleteCommand = new RelayCommand<IClosable>(DeleteMethod);
            Messenger.Default.Register<Member>(this, GetSelected);
        }

        /// <summary>
        /// The command that triggers saving the filled out member data.
        /// </summary>
        public ICommand UpdateCommand { get; private set; }

        /// <summary>
        /// The command that triggers removing the previously selected user.
        /// </summary>
        public ICommand DeleteCommand { get; private set; }

        /// <summary>
        /// Sends a valid member to the Main VM to replace at the selected index with, then closes the
change window.
        /// </summary>
        /// <param name="window">The window to close.</param>
        public void UpdateMethod(IClosable window)
        {
            try
            {
                // KEIRA: Messenger.Default.Send();
                if (window != null)
                {
                    var changeViewModelMessage = new MessageMember(EnteredFName, EnteredLName,
EnteredEmail, "Update");
                    Messenger.Default.Send(changeViewModelMessage); // sends "Update" message to
MainViewModel.ReceiveMember(MessageMember m)
                    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");
            }
        }

        /// <summary>
        /// Sends out a message to the initiate closing the change window.
        /// </summary>
        /// <param name="window">The window to close.</param>
        public void DeleteMethod(IClosable window)
        {
            if (window != null)
            {
                // TODO: Messenger.Default.Send();
                Messenger.Default.Send(new NotificationMessage("Delete")); // sends "Delete" message to
MainViewModel.ReceiveMessage(NotificationMessage msg)
                window.Close();
            }
        }

        /// <summary>
        /// Receives a member from the Main VM to auto-fill the change box with the currently selected
member.
        /// </summary>
        /// <param name="m">The member data to fill in.</param>
        ///
        // TODO: ChangeViewModel must receive the selectedMember from MainViewModel in order to
auto-fill input boxes.
        public void GetSelected(Member m)
        {
            EnteredFName = m.FirstName;
            EnteredLName = m.LastName;
            EnteredEmail = m.Email;
        }

        /// <summary>
        /// The currently entered first name in the change window.
        /// </summary>
        public string EnteredFName
        {
            get { return enteredFName; }
            set
            {
                enteredFName = value;
                RaisePropertyChanged("EnteredFName");
            }
        }

        /// <summary>
        /// The currently entered last name in the change window.
        /// </summary>
        public string EnteredLName
        {
            get { return enteredLName; }
            set
            {
                enteredLName = value;
                RaisePropertyChanged("EnteredLName");
            }
        }

        /// <summary>
        /// The currently entered e-mail in the change window.
        /// </summary>
        public string EnteredEmail
        {
            get { return enteredEmail; }
            set
            {
                enteredEmail = value;
                RaisePropertyChanged("EnteredEmail");
            }
        }
    }
}
```

MainViewModel.cs

```
using GalaSoft.MvvmLight;
using GalaSoft.MvvmLight.Command;
using GalaSoft.MvvmLight.Messaging;
using GymMembers.Model;
using GymMembers.View;
using System;
using System.Collections.ObjectModel;
using System.IO;
using System.Windows;
using System.Windows.Input;

// KEIRA: (AddWindow.xaml Pop-Up) Add namespaces.
using Prism.Mvvm;
using Prism.Commands;
using System.Windows.Data;

namespace GymMembers.ViewModel
{
    /// <summary>
    /// The VM for the main screen that shows the member list.
    /// </summary>
    public class MainViewModel : ViewModelBase
    {
        /// <summary>
        /// The list of registered members.
        /// </summary>
        private ObservableCollection<Member> members;

        /// <summary>
        /// The currently selected member.
        /// </summary>
        private Member selectedMember;

        /// <summary>
        /// The database that keeps track of saving and reading the registered members.
        /// </summary>
        private MemberDB database;

        /// <summary>
        /// Initializes a new instance of the MainViewModel class.
        /// </summary>
        public MainViewModel() //TODO: MainViewModel() constructor
        {
            members = new ObservableCollection<Member>();
            database = new MemberDB(members); // dependency injection of the members list into the
database instance
            members = database.GetMemberships();

            // KEIRA: (AddWindow.xaml Pop-Up) Attach AddCommand to AddMethod to act as an event.
            AddCommand = new RelayCommand<IClosable>(AddMethod);
            // KEIRA: (ExitCommand) Attach ExitCommand to ExitMethod() to act as an event.
            ExitCommand = new RelayCommand<IClosable>(ExitMethod);
            // KEIRA: (ChangeWindow Pop-Up) Attach ChangeCommand to ChangeMethod to act as an
event.
            ChangeCommand = new RelayCommand<IClosable>(ChangeMethod);

            Messenger.Default.Register<MessageMember>(this, ReceiveMember);
            Messenger.Default.Register<NotificationMessage>(this, ReceiveMessage);
        }

        /// <summary>
        /// The command that triggers adding a new member.
        /// </summary>
        ///
        // KEIRA: (AddWindow.xaml) Add AddCommand.
        public ICommand AddCommand { get; private set; }

        /// <summary>
        /// The command that triggers closing the main window.
        /// </summary>
        ///
        // KEIRA: (ExitCommand) Add ExitCommand.
        public ICommand ExitCommand { get; private set; }

        /// <summary>
        /// The command that triggers changing the membership.
        /// </summary>
        ///
        // KEIRA: (ChangeCommand) Add ChangeCommand.
        public ICommand ChangeCommand { get; private set; }

        /// <summary>
        /// The currently selected member in the list box.
        /// </summary>
        public Member SelectedMember
        {
            get { return selectedMember; }
            set
            {
                selectedMember = value;
                RaisePropertyChanged("SelectedMember");
            }
        }

        /// <summary>
        /// Shows a new add screen.
        /// </summary>
        ///
        // KEIRA: (AddWindow.xaml Pop-Up) Add AddMethod().
        public void AddMethod(IClosable window) // KEIRA: (Needs IClosable as a parameter to match
    RelayCommand/delegate signature.)
        {
            AddWindow add = new AddWindow();
            add.Show();
        }

        /// <summary>
        /// Closes the application.
        /// </summary>
        /// <param name="window">The window to close.</param>
        ///
        // KEIRA: (ExitCommand) Add ExitMethod().
        public void ExitMethod(IClosable window)
        {
            if (window != null)
            {
                window.Close();
            }
        }

        /// <summary>
        /// Opens the change window.
        /// </summary>
        ///
        // KEIRA: (ChangeWindow Pop-Up) Add ChangeMethod().
        public void ChangeMethod(IClosable window) // KEIRA: (Needs IClosable as a parameter to match
RelayCommand/delegate signature.)
        {
            if (SelectedMember != null)
            {
                ChangeWindow change = new ChangeWindow();
                change.Show();
                // TODO Send selectedMember to ChangeViewModel; ChangeViewModel must receive the
SelectedMember from MainViewModel in order to auto-fill input boxes.
                Messenger.Default.Send(SelectedMember);
            }
        }

        /// <summary>
        /// Gets a new member for the list.
        /// </summary>
        /// <param name="m">The member to add. The message denotes how it is added.
        /// "Update" replaces at the specified index, "Add" adds it to the list.</param>
        public void ReceiveMember(MessageMember m)
        {
            if (m.Message == "Update")
            {
                //TODO update
                int index = members.IndexOf(SelectedMember);
                members.RemoveAt(index);
                members.Add(new Member(m.FirstName, m.LastName, m.Email));
                database.SaveMemberships();
            }
            else if (m.Message == "Add")
            {
                members.Add(new Member(m.FirstName, m.LastName, m.Email));
                database.SaveMemberships();
            }
        }

        /// <summary>
        /// Gets text messages.
        /// </summary>
        /// <param name="msg">The received message. "Delete" means the currently selected member is
deleted.</param>
        public void ReceiveMessage(NotificationMessage msg)
        {
            if (msg.Notification == "Delete")
            {
                //TODO delete
                members.Remove(SelectedMember);
                database.SaveMemberships();
            }
        }

        /// <summary>
        /// The list of registered members.
        /// </summary>
        public ObservableCollection<Member> MemberList
        {
            get { return members; }
        }
    }
}
```

ViewModelLocator.cs

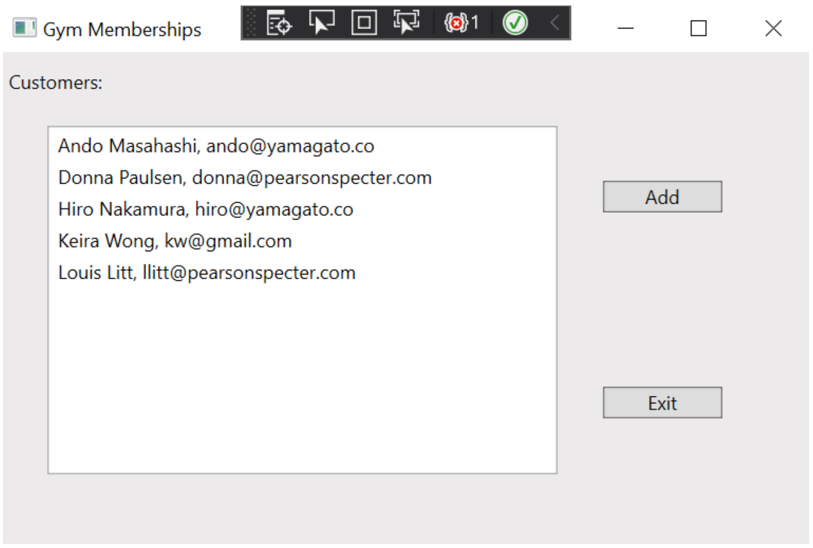
```
using GalaSoft.MvvmLight;
using GalaSoft.MvvmLight.Ioc;
using CommonServiceLocator;
using System;

namespace GymMembers.ViewModel
{
    /// <summary>
    /// This class contains static references to all the view models in the
    /// application and provides an entry point for the bindings.
    /// </summary>
    public class ViewModelLocator
    {
        /// <summary>
        /// Initializes a new instance of the ViewModelLocator class.
        /// </summary>
        public ViewModelLocator()
        {
            ServiceLocator.SetLocatorProvider(() => SimpleIoc.Default);
            SimpleIoc.Default.Register<MainViewModel>();
            //TODO: Register other views
        }

        /// <summary>
        /// A property that lets the main window connect with its View Model.
        /// </summary>
        public MainViewModel Main
        {
            get
            {
                return ServiceLocator.Current.GetInstance<MainViewModel>();
            }
        }

        public static void Cleanup()
        {
            // TODO Clear the ViewModels
        }
    }
}
```

IClosable.cs

```
namespace GymMembers
{
    /// <summary>
    /// An interface that lets objects be closed.
    /// </summary>
    public interface IClosable
    {
        /// <summary>
        /// Closes this object.
        /// </summary>
        void Close();
    }
}
```

OUTPUT

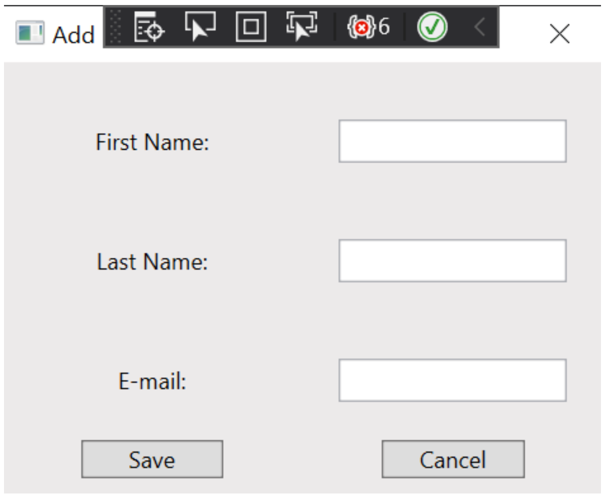
MainWindow

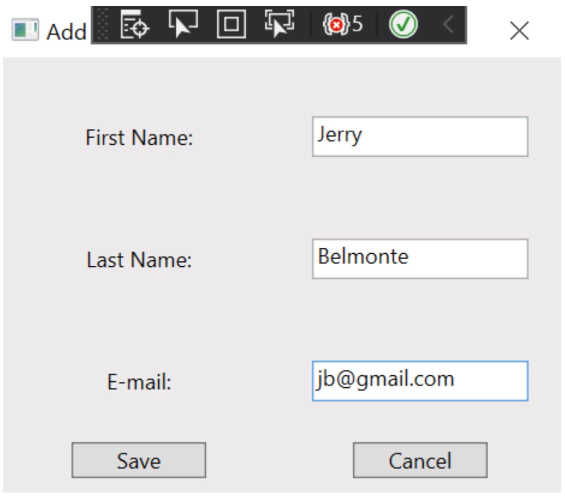


Customers:

Ando Masahashi, ando@yamagato.co
Donna Paulsen, donna@pearsonspecter.com
Hiro Nakamura, hiro@yamagato.co
Keira Wong, kw@gmail.com
Louis Litt, llitt@pearsonspecter.com

AddWindow



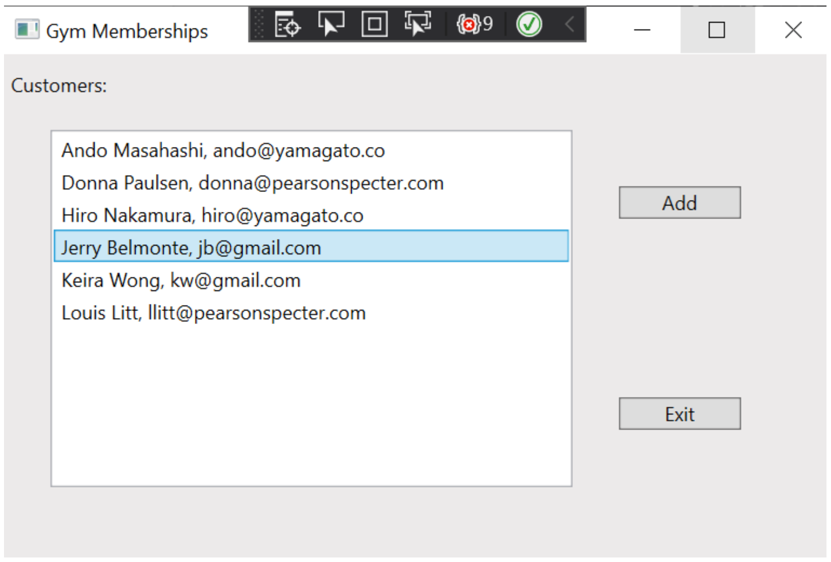


First Name: Jerry

Last Name: Belmonte

E-mail: jb@gmail.com

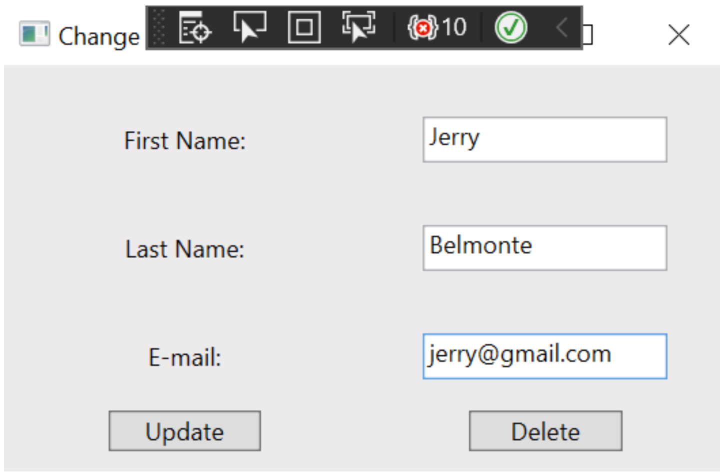
MainWindow (**AFTER add**)


Gym Memberships

Customers:

Ando Masahashi, ando@yamagato.co
Donna Paulsen, donna@pearsonspecter.com
Hiro Nakamura, hiro@yamagato.co
Jerry Belmonte, jb@gmail.com
Keira Wong, kw@gmail.com
Louis Litt, llitt@pearsonspecter.com

ChangeWindow

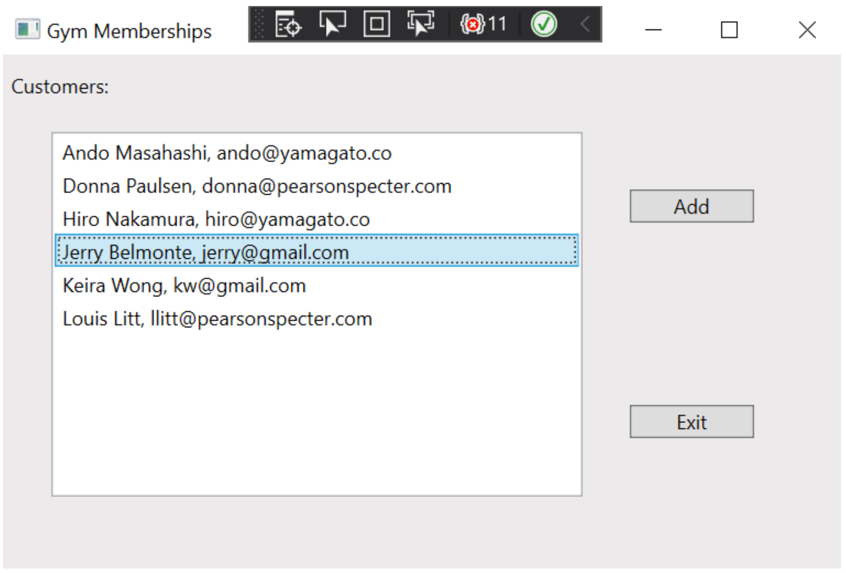

Change

First Name: Jerry

Last Name: Belmonte

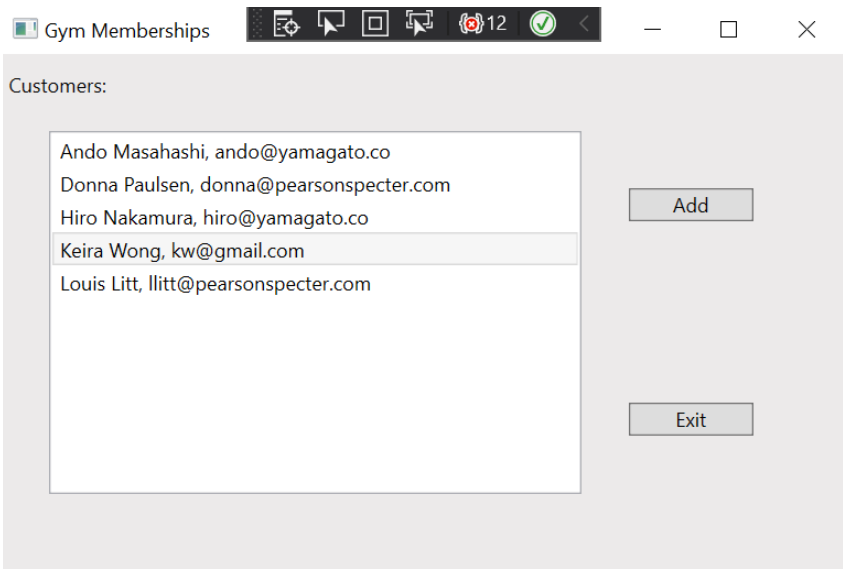
E-mail: jerry@gmail.com

MainWindow (**AFTER update**)



Ando Masahashi, ando@yamagato.co
Donna Paulsen, donna@pearsonspecter.com
Hiro Nakamura, hiro@yamagato.co
Jerry Belmonte, jerry@gmail.com
Keira Wong, kw@gmail.com
Louis Litt, llitt@pearsonspecter.com

MainWindow (**AFTER delete**)



Ando Masahashi, ando@yamagato.co
Donna Paulsen, donna@pearsonspecter.com
Hiro Nakamura, hiro@yamagato.co
Keira Wong, kw@gmail.com
Louis Litt, llitt@pearsonspecter.com

TEAM MEMBER’S WORK

Jerry Belmonte

- Member Database
- Add
- Delete / Update

Keira Wong

- Delete / Update
- Cancel / Exit