**Membership Maintenance**

1. **Code that uses indexers:**

*Below is the code used to get int value and get or set a particular member to the MembershipList class object-*

public Member this[int i]

{

get

{

if (i < 0 || i >= members.Count)

{

throw new ArgumentOutOfRangeException(i.ToString());

}

return members[i];

}

set { members[i] = value; }

}

*And below is the code to use an indexer:*

/// <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")

{

Member newMember = new Member(m.FirstName, m.LastName, m.Email);

if (members.Count > 0)

{

if (selectedMember != null)

{

database[members.IndexOf(selectedMember)] = newMember;

RaisePropertyChanged("UpdateMethod");

database.SaveMemberships();

}

}

}

else if (m.Message == "Add")

{

Member newMember = new Member(m.FirstName, m.LastName, m.Email);

database += newMember;

database.SaveMemberships();

}

}

1. **Code that overloads + operator and – operator. What code do you use to substitute those operator?**

*For delete:*

/// <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")

{

database -= SelectedMember;

database.SaveMemberships();

}

}

*For Add:*

/// <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")

{

Member newMember = new Member(m.FirstName, m.LastName, m.Email);

if (MemberList.Count > 0)

{

if (selectedMember != null)

{

MemberList[MemberList.IndexOf(selectedMember)] = newMember;

RaisePropertyChanged("UpdateMethod");

database.SaveMemberships();

}

}

}

else if (m.Message == "Add")

{

Member newMember = new Member(m.FirstName, m.LastName, m.Email);

database += newMember;

database.SaveMemberships();

}

}

*Defining operator overloading code to add:*

/// <summary>

/// Operator overloading with '+' sign, to add a Member to the list.

/// </summary>

/// <param name="m1">Object of MemberDB class.</param>

/// <param name="m">Member to be added to the list.</param>

/// <returns>Returns MemberDB object after adding 'm' to the list of members.</returns>

public static MembershipList operator +(MembershipList m1, Member m)

{

m1.Add(m);

return m1;

}

*Defining operator overloading code to Remove:*

/// <summary>

/// Operator overloading with '-' sign, to remove a Member to the list.

/// </summary>

/// <param name="m1">Object of MemberDB class.</param>

/// <param name="m">Member to be removed from the list.</param>

/// <returns>Returns MemberDB object after removing 'm' from the list of members.</returns>

public static MembershipList operator -(MembershipList m1, Member m)

{

m1.Remove(m);

return m1;

}

1. **Code to use the Changed Changed event to save the memberships and refresh the list box any time the list change:**

*Below is the code for to save memberships with new member in the file. To update listbox in MainView, we use RaisePropertyChanged(“UpdateMethod”) from class ObservableObject:*

database[members.IndexOf(selectedMember)] = newMember;

RaisePropertyChanged("UpdateMethod");

database.SaveMemberships();

1. **Code to wire the event handler to the event for the Save:**

*We use Messenger class which allows objects to exchange messages to interact. We use below code to register MainViewModel.cs file to register for messages with MessageMember class objects. When changes are made to the list of members in any other windows (like AddView, ChangeView or DeleteView), Message is been fired.*

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

1. **Code to open the Add form:**

*To create a RelayCommand object and passing a method which will get open add window. Below is the code:*

AddCommand = new RelayCommand(AddMethod);

public void AddMethod()

{

AddMember add = new AddMember();

add.Show();

}

1. **Code to transfer data from the Add view to the Membership Maintenance view**

*Below is the code where we send a notification using Messenger object of Galasoft.MvvmLight.Messaging package:*

MessageMember newMessage = new MessageMember(EnteredFName, EnteredLName, EnteredEmail, "Add");

Messenger.Default.Send(newMessage);

window.Close();

And above MessageMember class object is used to pass with Messenger object.

MessageMember consists of two variables. String message and Member object, which we need to pass.