Review

To sk222sw

The class diagram is correct but a bit cluttered and unstructured which makes it hard to read at first sight. When you look into It, it makes sense. Maybe a different kind of UML program would make it easier to read.

The sequence diagrams is too small and lack some information and steps of the sequence are left out. For example in Add member sequence diagram before AddMember is called there are: MainView.ShowMenu(), MainView.GetMenuChoice() and so on. "The elements shown in SSDs (operation name, parameters, return data) are terse. These may need proper explanation so that during design it is clear what is coming in and going out." Larman C. Chapter 10.8

At the start of the controller you use enumeration to handle the input from the user. I think this is a good solution to encapsulate dependency between the service class and the controller. However this is solution is lost further down in the controller. The controller start to print text to user which the view should do. Also the controller starts to read inputs from the console. This is not correct and it's not a model-view-controller separation. "In addition to dividing the application into three kinds of components, the model–view–controller design defines the interactions between them.[13]

- A **controller** can send commands to the model to update the model's state (e.g., editing a document). It can also send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document).
- A **model** stores data that is retrieved according to commands from the controller and displayed in the view.
- A **view** generates an output presentation to the user based on changes in the model.

" Wikipedia under 2.2 interactions.

Also dependency are not encapsulated when the user is going to select 1 for compact or 2 for verbose. Use enumeration the same way as when you wanted to add a member or view a member. Then later down in the controller you use model-view separation when the user is managing members.

MemberListView reads data in SelectMember which is a domain logic and should be placed in a model class. The data should be passed through the controller like in the other methods in that class.

MemberView ChangeMember is changing the domain state when saving the member. This is what the controller should do. In the AddMember method you did it correct.

Strong points

The application classes of the application have good cohesion and are not too big. The separation between classes makes it easy to understand their function. Some parts of the application have a good model-view-controller separation.

Weak points

Some parts of the application have wrongful model-view separation.

I think the application passed 2 grade.

Refrences:

Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0-13-148906-2

https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller