Peer Review for Andreas Anemyr

Test of the runnable application.
Boat CRUD is not implemented, throws unhandled exceptions.
View member is not implemented, throws unhandled exceptions.
Quitting the application does not work.

Test of compiling the project.
Compiling the project worked.

Implementation vs Diagram
The diagram only shows inheritance. We would like to have associations and dependancies shown aswell as is shown in the book [1, p380, ch16.1, f16.1]. The sequence diagram is good.

Is the architecture ok?
Overall the architecture is good. Classes has good cohesion [1, p432, ch17.8], one controller for each view is a good amount. We are missing a package for the Cache-class though. There is a clear separation of the model and the view and the controller-pattern is used in a good way [1, p431, ch 17.8].

Is the requirement of a unique member id correctly done?
Member does not seem to have an ID-property. As of now, it seems like the unique identifier for each member is the social security number. I guess this kinda solves it, but still not really.

Quality of the implementation/source code
Well, everything is not implemented but the things that are implemented seems good. It feels like you have tried to do a very complex structure that focuses on reusability. It was pretty hard though to read and navigate the code, especially since the class diagram didnt have any associations.

Mixing of PascalCase and camelCase should be eliminated. Chose one and stick with it (PascalCase for C# tbh).

Is interfacing an abstract class a good idea? What does it bring to the table? We think the abstract class would suffice.

What is the quality of the design? Is it object oriented?
The quality of the sequence diagram is good, it shows what is going on in a good way. As stated before the class diagram needs more arrows (dependancy/associations).

As a developer would the diagrams help you and why/why not?
It feels like we are repeating ourselves, but the sequence diagram helps us understand what is going on in the application. It would be better to have the diagram show a specific task, like is shown in the book [1, p351, ch351]. The instructions said: One diagram for output, one diagram for input.

What are the strong points of the design/implementation, what do you think is really good an why?
As we said earlier, this solution feel complex(in a good way), we like your ambition. Sorry that you couldnt find time to implement everything because the things that are implemented looks very good. Code-reading for us was hard because of your ambition-level, not because your code was bad.

Many enum-types are good, we like enums! Instead of hardcoded strings.

As stated earlier, we thing cohesion is good, number of classes and their responsibilites are "lagom".

What are the weaknesses of the design/implementation, what do you think should be changed and why?
As stated before, more arrows in the class diagram. Implement methods that throw NotImplementedException. Feels like we are repeating ourselves but these question are pretty DRY.

Do you think the design/implementation has passed the grade 2 criteria?
Sadly we have to say no, since the application is not fully implemented yet. If it was, this workshop would definetly pass.

References

Larman C., Applying UML and Patterns 3rd Ed.
Please note that the page numbers referenced in this document is from the PDF-version, not the hardback copy. Link to the pdf:
https://drive.google.com/file/d/0B3-OzxP8Wm1IbmhhTGhIMXlQYlU/view?usp=sharing