Peer Review 1 of ph222md

Test the runnable version of the application in a realistic way. Note any problems/bugs. / Try to compile/use the source code using the instructions provided. Can you get it up and running? Is anything problematic? Are there steps missing or assumptions made?

- Goes in an infinite loop when looking at a member's information when no members are added, cannot exit from the loop.
- Have to redo from start when typing in wrong format in the personal number and not just redo the personal number.
- “Add something else” is misleading.
- 2 people can have the same personal number
- After deleting a member the program start acting weird and asking about personal number.
- Cannot change/delete a boat without an ID and we are not being able to obtain the ID of the boat.
- Cannot exit any of the loops if you would change your mind and don't want to delete any boats anymore, except for the add boat where you can go back by writing “q”.

Does the implementation and diagrams conform (do they show the same thing)? Are there any missing relations? Relations in the wrong direction? Wrong relations? Correct UML notation?

- Missing several diagrams(delete member, add boat, delete boat, verbose list...)
- Associations could be added to the class diagram to help understand how everything work together.[1]
- The class diagram should show dependencies[2]
- In the add member and the Compact List diagram the “reg.” part is unnecessary since it is understood when it goes to the next class in the diagram and“:Register” is not the same name as the corresponding class in the code which is “BoatClubRegistry”.
- The add member diagram does not include getters and setters[3], a “printMemberToMemberFile” if “membersRegistered = 0”.

---

[1] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 16.4 Ways to show UML Attributes: Attribute Text and Association Lines

[2] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 16.11 Dependency

[3] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 15.4. Basic Sequence Diagram Notation

- The Compact List diagram does not include getters and setters[4]. And The Compact List does not show “member.getBoatList().size()” which makes a call all the way to the boat class.
- The Compact List diagram does never call the “LoadBoatFile()” method but loads all the boats in the “LoadMemberFile()” by creating a new boat object in a loop for every boat in the file listed, thus a loop with “boat()” method and also all the other dependencies that are in there should be next to the “LoadMemberFile()” and the other one removed.[5]

Is the Architecture ok?

- There is a good model view separation but it is missing the controller and is instead handling the user inputs in view package which is not recommended unless it is only low-level inputs.[6]

Is the requirement of a unique member id correctly done?

- Yes it is, We didn’t get any member with the same id during our run of the program.

What is the quality of the implementation/source code?

- Not so clean code since a lot of the code in model package does not have anything to do with the functionality but with the behavior depending on what the user puts in and that should be in a separate package called controller.(“checkAlternativeInTwoWays()” for example).[7]
- good comments and easy to follow, but some of the comments are more or less “obvious” and could removed to clean up the code in our opinion.(// This will loop as long there is something in the file while (scan.hasNext()) For example)
- Good names.
- Good separation of the classes, making them independent as much as possible.(not object calls back and forth).
- No duplicates or deadcode

What is the quality of the design? Is it Object Oriented?

- It is good object oriented. There are no calls being made to attribute in other classes without an object attached, it is done only through object calls using getters and setters.
- Boat types could be individual classes extending an abstract class called boat for a more object oriented design.

---

[4] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 15.4. Basic Sequence Diagram Notation

[5] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 15.4. Basic Sequence Diagram Notation

[6] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 13.7.Guideline: The Model-View Separation Principle

[7] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 13.7.Guideline: The Model-View Separation Principle

- As mentioned above it does not provide a clean look of the pure functionality of the program as it is mixed with the user input handling.[8]
- The model package has a dependency on the view package through compact and verbose list methods since they create a “Console” object and use it. The things in view package is related to the interface which you should be able to change without having to make changes in model package code as well.[9]

As a developer would the diagrams help you and why/why not?

- The diagrams are not complete but with what is done we think it is easy to follow with it through the code.

What are the strong points of the design/implementation, what do you think is really good and why?

- Easy to follow.
- Good comments

What are the weaknesses of the design/implementation, what do you think should be changed and why?

- The architecture is fine but mixing the functionality and the user handling makes it more messier than it have to, to look at for someone who have not worked with it before.
- The model package has a dependency to the view package.

Do you think the design/implementation has passed the grade 2 criteria?

- Even tho not perfect, you can follow through the code and understand well using the comments.
- Using the interface was relative smooth with some flaws.
- Storing and loading information works
- The diagrams are not finished and the interface has minor flaws, but other than that we believe it passed the grade 2.
- There are no runnable version of the program as it was a requirement to have and not just compile the source code, recommended is exporting the project as .jar file through eclipse or other editor.

---

[8] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 13.7.Guideline: The Model-View Separation Principle

[9] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 13.6. Guideline: Design with Layers, Chapter 13.7.Guideline: The Model-View Separation Principle