Peer review for Ola Franzén

Peer review by: Jakob Brandt jb223df@student.lnu.se, Mauro J. Pappaterra mp222tb@student.lnu.se, Thorvald Gudjonsson tg222eu@student.lnu.se

Architecture:

The model appears to be coupled to the user interface correctly. Some of the methods in the Model return Booleans, this is used for UI purposes and we don’t think it’s ideal, as stated on the book **(1)**. None the less most domain rules are included in the Model and not in the UI, this is a good practice according to Larmans book **(2)**.

The unique Member ID is handled properly as a randomly generated 8 character String, it does not follow any particular patron or provides with any information about the member, and nonetheless we think that it was a good implementation to solve the task.

Code:

The standard of the code is good, as the program works out of the box and without any big issues or bugs. The naming of the classes, fields and methods are correctly implemented as they help to illustrate their roles and functions in the code. There are no duplicates or dead code found in the program.

Design:

The design of the program is Object Oriented and is properly implemented as such. We also think that this group did a good job with the implementation of the MCV pattern as the coupling the model, control and view is correct.

We did find some flaws in the design. There is not proper error handling for member duplication, there is no proper encapsulation for the social security number, this, assuming it is a Swedish id number, should be encapsulated to avoid an invalid id number to be entered. If not a Swedish number it should also be encapsulated according to the pattern that that particular id number would follow. In practise it won’t be too difficult for a user to mistype a 10 digit string of numbers, as this is a common error, and the system should be designed to prevent this.

A big flaw that we find is that a field corresponding to the ‘Surname’ is missing in the member class, we think that this will definitely be important when there’s a large number of members in the database. Moreover there is no error handling for members duplicated names and identification numbers, so we can actually have several duplicated members with the same name and the very same id number but with a different member number, or two different members with different names but the same identification number.

We also spotted some flaws in the design of the menu prompts. For starters viewing a whole list of members and selecting them by number when ‘editing’ or ‘deleting’ members could become rather a problem when there’s a large database in the system; We think that ‘adding a boat’ should be presented in the main menu, as now it appears to be ‘hiding’ as a submenu of ‘view/change/delete member’; At the start of the program we read in the console that we can press ‘b’ at any time to go back in the menu, when ‘b’ actually returns to the main menu every time it is pressed no matter where in the menu we are; After adding a new boat the menu prompts you to ‘press ‘a’ to add a new member or ‘b’ to go back to the main menu’. This message also shows when finishing with the edition or deleting a boat, we think that it should exit to member edition menu, this would be more practical in case that the user wants for example, to register more than one boat at the same time or in any case it should exit to the main menu. The same message shows after printing the list of member in a verbose display. We are not sure if this is a bug or a design mistake, but we think it will be confusing for the user as it does not follow a coherent workflow of actions.

We don’t think the diagrams are correct or helpful as some methods appear as classes and referred as enumerations, we think they have misunderstood some of the concepts behind it. Moreover interface association among model classes are not a good practice **(3)**. The interactions diagram seem to be correct and thoroughly documented.

All in all the program works perfectly and all requirements are met. Nonetheless we really think that some brushing up is necessary. The main weakness of this design is the lack of proper error handling and encapsulation. The strong point is a well define boundaries in the implementation of the MCV pattern. We do believe that the program is good and deserves a passing grade.

References:

1. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 17, page 444
2. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 32, page 692.
3. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 16.