Workshop 2 – Peer review 1

Review Clarity
For documentation/diagrams
In your documentation and diagrams you are using English as required and does so with only one spelling error. This being:
1. In the sequence diagram ”LookupMembersInfo.pdf”, the method name printSpesificMemberUi. Spesific should be Specific. (Should be changed in the source code also)

Your diagrams uses the correct UML for design. Unfortunately I could not open the following UML diagrams:
1. CreateMember.pdf.
2. DeleteBoat.pdf.
3. DeleteMember.pdf.
4. ShowList.pdf.
The other .pdf-files, not mentioned above, opened fine and at least one of them is a sequence diagram for input and one of them is a sequence diagram for output. The requirement being to provide sequence diagrams where one is for input and one is for output makes you pass this requirement.

The UML diagrams are readable, has a good layout and uses a consequent format for names. Good! The only thing I can remark on is the naming ”ui:view::Console” and ”boatClub_model::BoatClub”, I would suggest to change it to ”view::Console” and ”model::BoatClub”, to easier understand where in the code one could look for more details.

The naming of classes follow the rules of thumb, being a noun in singular form. Associations have full role description and uses multiplicity where it is needed.

For code
Indentation and formatting of blocks is mostly correct, but unfortunately not everywhere in the source code (Martin, p.88, 2008). Places where the indentation/formatting should be fixed:
1. Console.java:
    1. printRegisterBoatUi() – Function name and first block is indented with 1 tab + 1 space. Function closing bracket is indented 1 tab.

    2. printDeleteBoatUi() – Function name and first block is indented with 1 tab + 1 space. printFooterMessage() is also indented as above.

    3. printUpdateBoatInfoUi() – Function name and first block is indented with 1 tab + 1 space. printFooterMessage() is also indented as above.

    4. printSpesificMemberUi() – Function name and first block + if-statement is indented with 1 tab + 1 space.

    5. printFooterMessage() – Whole function indented with 1 tab + 1 space.

2. BoatClub.java:
    1. findBoatIndex() – for-loop closing bracket is not indented and could be misstaken for the functions closing bracket.

    2. isMemIdExists() and isBoatIdExists() – missing brackets in if-else-statement.

3. DataIO.java:
    1. writeMemberDatatoFile() – Indentation is off. Try-block is indented 1 tab + 1 space. Catch-block is indented 3 spaces.

The naming of classes/operations/attributes and arguments are consequent and uses real words. Only remark is that DataIO.java, writeMemberDatatoFile should be changed to writeMemberDataToFile with a capital T on ”to” (Wikipedia, 2017). Personally I would also rename the BoatClub.java methods isMemIdExists and isBoatIdExists to doesMemIdExist and doesBoatIdExist, this would make the code read more like a book. In the Console.java you should change printSpesificMemberUi, Spesific should be Specific.

Generally the code is readable and understandable, even for someone like me who haven’t programmed in Java myself.

The comments that are present in the source code all feel redundant, since the naming of methods, variables and whatnot is self explanatory (Martin, p. 55, 2008). This is a good thing! I think you should get rid of all your comments, since they do not contribute anything new to the code, your code is already explaning itself. However, one place that I would like you to add a comment is in DataIO.java method writeMemberDatatoFile, the catch-block. Why is it empty? Is it supposed to be empty?

Please look over all your methods to see if there is whitespace that can be removed. In some places there are alot of whitespace that can make it harder to see where a method ends (Martin, p. 80, 2008).

Another tip is to order your methods based on when they are called (Martin, p. 84, 2008).

Overall I think the clarity is fine as it is now, the things I’ve remarked on is really just nitty gritty things that would make what is already good even better.

Review Completeness
Your hand-in contains all required parts of the workshop. Good!
I was able to run the system and tested different kinds of scenarios on it to make sure it met the listed requirements. These are my remarks on the system:

1. Menu – If I write something other than a number, the program ends and has to be restarted. Not user friendly.

2. Create a member – I can save a member without typing in a name or personal number.

3. Create a member – I tried name: Test Testson and personal number: 010101-0101, but the program ended and had to be restarted.

4. Create a member – I tried name: Sofia K and personal number: 9101270000, but the program ended and had to be restarted.

5. Create a member – I could save a member using the same name and personal number as another already existing member.

6. Delete member – If one writes letters instead of numbers, the program ends and has to be restarted.

7. Update member info – Can update a member without writing in a new name, making the system save a member without a name.

8. Register boat – Can save a boat without specifying the boat type.

9. Change boat info – Can update a boat without specifying the boat type.

10. Verbose List – Only shows name, member id and boats with boat information. Missing personal number.

11. Exit – Shows the menu once more before closing.

I suggest you implement some simple error handling to make sure the program does not end and has to be restarted over and over again. Although it is not required to be a user friendly application, this is only a suggestion and nothing that you have to do anything about if you do not want to.

You should however implement error handling for:
1. Empty input field. User has to provide for example a name, boat type etc.
2. Show the user what kind of input you are looking for when one is supposed to create a new member. For example: Please enter personal number (yymmdd-xxxx):, or however you want the personal number to be written.
3. User should not be able to create an already existing member.

Other: Add personal number to the Verbose List to meet the requirements.

Review Content
There is a model-view separation, but in one of the model classes there is a System.out.println. (See BoatClub.java, BoatClub catch-block). This print out should be handled by the Console class, maybe in a try-catch-block, where the BoatClub class throws the exception and the Console class catches it and prints it out. This to make the model-view separation strict and to allow easy porting of the model layer to another user interface (Larman, chapter 13, guideline 13.7, 2004).

The design and implementation is object oriented. The classes have high cohesion, since they all focus on what they should be doing. They also have low coupling, since they are not too connected to other entities. Changes in one class therefore does not make a huge impact on the other classes.

You do not use any static variables or operations, and no global variables. As far as I can see you do not have any hidden dependencies, which could produce unexpected side-effects. Good!

Informations are encapsulated by the lower layers in the sense that the UI for example does not show the nitty gritty details on what the model does when a user wants to create a new member.

You are using a natural design inspired by the domain model.

Lastly your design artifacts are in sync with the code.

Reference
Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.* Boston: Addison Wesley Professional.

Wikipedia. (2017). *Naming convention (programming).* Retrieved 2017-10-04, from https://en.wikipedia.org/wiki/Naming_convention_(programming)#Java.

Martin, R. C. (2008). *Clean Code A Handbook of Agile Software Craftsmanship.* Boston: Pearson Education, Inc.