

Peer review comments Tobias Johansson

Songho Lee and Sarpreet Singh Buttar

General comment

It would be easier for reviewers if you had your model classes and view class in separate folders.

Class diagram (Architecture)

- Arrow from console to MemberDatabase presents wrong information. It should be replaced to direct dependency.
- Console class, which we assume program.cs in your case, has direct dependencies on Member(Line 92)
- Arrows are incorrectly presented [1].

Model

- d. **[Boat]** has a length in integers. By the way in which unit the input is presented? Meters? Shouldn't it be possible to handle 3.2 meters in such case?

```
Enter length of the boat:
5.2
Indatasträngen hade ett felaktigt format.
Programme terminated despite I presented normal value for length.
```

- e. toString() of boat class is against of model-view separation principle, because presenting those in string or whatever thing is view's responsibility, not from the model side.
- g. Getters and setters of a boat class are written in private. Therefore, we think it is not accessible from member class. Since we are Java programmers, you might want to ask someone else who fluent with C#, but when we were reviewing other programme which is also written in C#, getters and setter were public.
- h. **[Member]** has a static variable 'theCounter', which is very against of object oriented principle. Besides, all fields inside the Member class are public. Thus, it leads to a conclusion that there is no encapsulation.
- i. For updateBoat method we wonder if it is necessary to overwrite the existing boat with new boat object. It could be handled by its getters and setters.
For update/delete boat you are presenting an index of a boat to the method. This is not fulfilling the requirement 12 that objects should connected using association. Therefore, we recommend to take Boat as an argument(parameter), instead of indexes.
Again, toString should be avoided.
- j. **[MemberDatabase]** is also using its create member, update member, delete member method with integer id. It is recommended to avoid using integer ids, similar to above mentioned reason. Moreover, all the methods are accessing directly the fields of member class which is also against OO and encapsulation. It would be nice if getter and setter can be used to avoid these problems.
In addition, you are printing out the members in list members() which is violating the MV separation. Move it to view side.

View

- k. Separate view class from the main console programme, so that you can follow the model view principle.
- l. Programme crashes with erroneous input (char instead of number). Is it an intended design that to close the application once it has a wrong input? We suppose this is not fulfilling the requirement 13 of workshop 2

```
C:\WINDOWS\system32\cmd.exe
Enter length of the boat:
p
Indatasträngen hade ett felaktigt format.
C:\Users\songhokun>
```

- m. Delete a member if a member does not exist. we expected an error here.
- n. Register and update boat method has code duplications.
- o. Reading a file creates a member. Couldn't it be in the model side?
- p. Look up a specific number: we receive numbers next to name. What does it stand for?
- q. Compact / verbose list: we have no idea what those numbers mean.

```
Kommandotolken - "H:\Syncdocs\arbetar\peerreview\Tobias Johansson\\"
Enter member id:
1
Johan 1 0

Johan Strauss 1

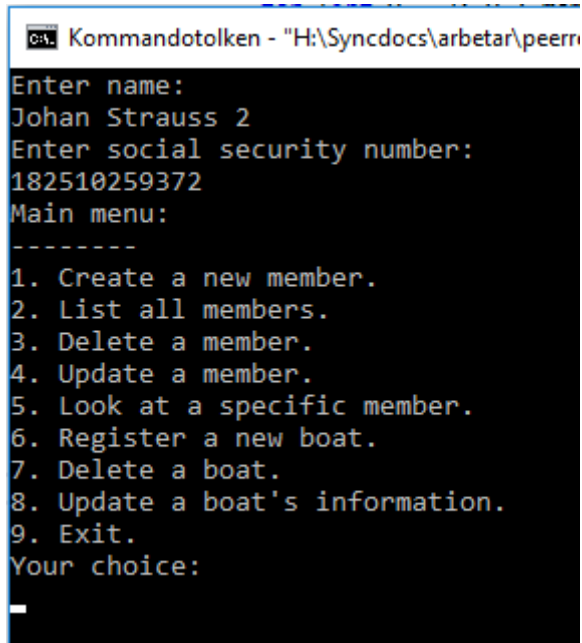
Main menu:
-----
1. Create a new member.
2. List all members.
3. Delete a member.
4. Update a member.
5. Look at a specific member.
6. Register a new boat.
7. Delete a boat.
8. Update a boat's information.
9. Exit.
Your choice:
I do not understand what those numbers stand for.
```

- r. Update a boat's information: how am I supposed to know index of the boat?

```
Välj file://SONGHOHEM/homes/songhokun/Syncdocs/arbe
Enter member id:
1
Enter index of the boat:

```

- s. I am afraid that using whitespace to separate fields in the file is not a good idea. The example below provides perfectly normal input to the programme.



- t.
- u. However, this somewhat leads to erroneous situation.

Summary

Are there steps missing or assumptions made?

Yes, length is taken in integers.

Does the implementation and diagrams conform (do they show the same thing)?

No. Relations are not presented in correct associations. Plus, association from program.cs and Member is missing. Furthermore, strictly saying there is no 'console' presented but only program exists, which is combination of main programme and view class.

For sequence diagram, we assume that first one which is named as input, seems to be okay, yet we wonder where are the other diagrams for updating, deleting members and boats and other requirements.

Is architecture ok?

Since there is no strict model-view separation, especially the part that literally prints out on the CONSOLE(!) on MemberDatabase.cs (line 58), unfortunately we cannot say that it has an okay architecture. Besides toString() on boat and member makes it too much coupled on specific UI.

Is the requirement of a unique member id correctly done?

It is fulfilled, but we would say not in a way according to the requirement (due to static)

What is the quality of the implementation/source code?

Due to bad encapsulations, duplication of codes (look at registerBoat and updateBoat in program.cs), and index dependencies instead of using object, we concluded the code does not have ok quality.

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

It is not object oriented due to many hidden dependencies, and using of keys/ids instead of referring to object itself. Besides, it is depending on static variables.

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

As a developer it would be difficult to implement programme based on the given diagram as it lacks some associations and it contains confusing association, such as realization arrow from console to Memberdatabase.

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

Usage of BoatType as an enumeration is a plus point in the design since it provides only limited choices.

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

This question is answered above.

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

Due to above mentioned reasons on this review, we would recommend to revise the programme in order to pass grade 2 criteria. Some of requirements are not implemented, such as error handling. Besides, when I tried to test the program it could not register a boat, which is assumably due to there is no separation between member data and boat data when the name contains spaces. We may recommend to use different separator between member information and boat information other than whitespace.

Works Cited

- [1] D. Bell, "The class diagram," IBM, 15 September 2004. [Online]. Available: <http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/>. [Använd 10 October 2016].