# Peer-Review

## Workshop 2

### Peer-review on Rebecca Fransson, Hampus Jarleborn, Rickard Söderman

***Test the runnable version of the application in a realistic way. Note any problems/bugs.***

I downloaded the executable from the provided link and tried to run it but it fail. I got an unhandled exception, and then the console window closed down. The program is working if I build it from Visual Studio. I figured out that the program is dependent on the bin file that saves the member data (‘member.bin’) and if this file does not exists, the program crashes.  
  
In method List<Member> in class BoatClubDAL, the file streamer tries to read a file, but if the file does not exist, there is an exception thrown. A better way to do this, would be to check if the file exists before the streamer tries to read the file. If the file doesn’t exist, an empty List<Member> is returned. An alternative would be to throw an exception, but this exception needs to be caught in the ‘view’, which is not the case now.   
(File.Exists(): https://msdn.microsoft.com/en-us/library/system.io.file.exists(v=vs.110).aspx)  
  
Furthermore, Exception are thrown in the model and displayed in the view with consoleView.displayExeptions(e.ToString()). e.ToString gives a long message, and is used for developers, not users of a program. e.Message displays a more appropriate message. (consoleView.displayExeptions(e.Message)).

The program throws an exception when I use dot instead of comma (for instance inputting boat length). This can be handled by the class and if input contains a dot, this converts to comma, before the input is converted to a float number).

***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?***

I missed the instruction that both files where needed for the program to be able to run. Other than that, there were no problems finding source code and diagrams in the repository.

***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?***The class diagram shows association to List<model.Boat> and List<model.Member>. ‘List’ is by definition objects (or classes) but it belongs to .Net framework and is ‘external objects’. In my opinion, I don’t think it is necessary to show dependencies to these (the objects could have been saved in any collection, and that’s not so important, that it needs to be shown). Instead it would be more correct to show that Members and Boat has dependencies. (Secretary and Member already shows a dependency).   
In the diagram, the enum <<BoatType>> shows an association to boat. But in your code, the enum is nested within the class and is part of the class (The enum is actually not BoatType, but Boat.BoatType.). Enums does not need to be declared inside a class, but could be if desired. This is a design decision you have to make.

Is the Architecture ok?

* ***Is there a model view separation?***
* Yes, it has a model-view separation.
* ***Is the model coupled to the user interface?***
* No, the model is not coupled to UI.
* ***Is the model specialized for a certain kind of IU (for example returning formated strings to be printed)***

The model throws exception messages that is caught in the view. These messages are not formatted in any way in the model.

* ***Are there domain rules in the UI?***

No, there are no domain rules in the UI.

***Is the requirement of a unique member id correctly done?***The unique id is generated as a random number in the controller, and before assigning the id, it is checked against all other members, so that it is unique. If it is not, a new randomized number is generated and checked.

I think this is in the border to violate the MVC-principle. To generate a unique id, and to check if it is unique could have been done in a model class. The controller should not have that kind of responsibility.

***What is the quality of the implementation/source code?***

* ***Naming***You have good naming. It is easy to understand what the methods and variables do, by just reading the name. The only naming I don’t agree to, is Secretary. As I know about the problem description in workshop 1, I know that there is a secretary involved. However this is a ‘role’ and should not be a name of the controller. Let say that workshop 5 is an extension to this workshop, and the assignment is to create a ‘secretary’-role (similar to member, but with different privileges. Then there will be a naming conflict between your controller and the secretary model.
* ***Duplication***It has not much duplication. There is some in the view, but that is difficult to avoid.
* ***Dead Code***I have not found any dead code.

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

* ***Objects are connected using associations and not with keys/ids.***Yes, Associations are used to connect objects.
* ***Is GRASP used correctly?***GRASP principle is an aid for using the design principles ‘pattern of assigning responsibility’. Responsibilities can be of two types: *Knowing or Doing.* This means that the object can either do something itself, initiate some other object to do something or controlling actions in other objects. Knowing responsibilities means that the object knows about private encapsulated data, knowing about related objects or knowing about things it can obtain or compute[[1]](#footnote-1).   
  I will use examples to illustrate GRASP implementation in your application:

CREATOR:

Secretary has the responsibility to create new Member. This follows the creator pattern, since Secretary has the initialization data for Member.

LOW COUPLING:

I think you have quite low coupling between the classes in the model, and high coupling between the controller and the models (which is natural).

HIGH COHESION  
The classes Member and Boat has high cohesion, while the controller and the view has low. One way to accomplish higher cohesion in the view would be to spit this class into more classes, one for the menu, one for input and one for displaying outputs.

CONTROLLER:

You are using the controller pattern.

* ***Avoid the use of static variables or operations as well as global variables.***

No static variables or operations are used, nor are any global variables used.

***Information should be encapsulated.***

You could do some more work on encapsulation. The Secretary class has lots of public methods that is only used inside this class. These could instead be private.

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

The class diagram is simple, and this makes it easy to read and understand. Yes I would be helped to with the diagram.

***What are the strong points of the design/implementation, what do you think is really good and why?***There are low coupling and high cohesion in the model classes.

***What are the weaknesses of the design/implementation, what do you think should be changed and why?***The exception handling are not correct implemented. In my opinion, the controller is in the borderline of having too much responsibility that could be handled by the model. (I’m thinking primarily of the assignment and checking of unique id)

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

Yes I think it has passed grade 2 criteria.

1. Larman, C (2005). *Applying UML and Patterns (Third Edition)*. Upper Saddle River: Prentice-Hall. Chapter 17 GRASP: Designing Objects with Responsibility. [↑](#footnote-ref-1)