# Peer review Workshop 2

For: Johnny Pesola (jp222px), Matilda Ingman (mi222ep), Johanna Larsson Cornejo (jc222fi)  
By: Ola Franzén (of222au)

### Testing the application

The program runs mostly well without any crashes, and it is easy to navigate and do the actions you need to do. I only could find a small report error when changing name or personal number for a member. When you have filled this in you get an error message that there already exists a member with that personal number (even if the name is changed or personal number is changed to an unused number). However the change seems to go through so this is probably only a faulty error report from the program. Other than that it seems to run without any obvious bugs.

### Source code

I could not compile the source code because that I only have Visual Studio 2010 installed on this machine (which doesn’t support Net Framework 4.5 as used in this project), so I will assume it would compile on a later VS version.

Looking through the code the naming of functions, fields and classes are done well and in a self-explanatory way. There also seem to be mostly no unnecessary duplication of code or any dead code. However I could argue that there are some possible reduction of code possible in the MemberDAL and BoatDAL classes, where similar code is used in many functions, especially the Get() and GetAll() functions in MemberDAL and GetBoats() and GetBoat in BoatDAL. I did also found a few methods in the MemberService class which seems unused (and not fully implemented); IsBoatLenghtValid(), IsMemberNameValid() and IsMemberPersonNumberValid().

Other than that the program also has well used exception handling.

### Architecture

The design of the program is object oriented and also well divided by the MVC pattern, with the different responsibilities handled in the different layers as Larman describes [1, 209]. No view specific info is found in the models and no domain rules are found in the view.

The creation of the member id is handled by the database in the form of a primary key. However it seems this id is not presented in the member lists when using the program, as stated in the second requirement.

### Quality of design

One concern I have is that in some functions the association to model classes is done by keys/id, for instance in the MemberService model class, the functions GetMember(int memberId) and DeleteMember(int memberId). A better way would be to only reference by association to the objects, and let the DAL classes handle the keys/id connected with the row in the database.

Other than that I think the quality of code is good, and the GRASP principles are well complied. The classes have high cohesion and not too large. For example the access to the database is clearly separated into the different DAL model classes. At the same time the classes also have low coupling as described by Larman [1, 299].

A small concern I have is that some static variables are defined in the MenuView, which should and probably easily could be avoided.

### Diagrams

The diagrams do help to understand the layout of the program. The class diagram is well conformed and shows a lot of info. It seems to miss the DALExtensions class though. The sequence diagrams are well documented and shows all the actions and information handled.

### Summary

Overall I think this is a well performed design and implementation that solves the task and requirements. A few smaller concerns have been spotted as described above.

The best point in this project I think is the general arrangement of the classes and how it easily can be understood and used.

The biggest weakness might be how some functions handle the associations to model classes with id instead of an association to an object.

I think this design and implementation has passed the second grade.

## References

1. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062