Daniel Blixt
Daniel Nilsson

2015-10-08

Peer Review - Workshop 2

For: Andreas Bom

We tested the binary version of the program and found it to be pretty good, it fulfills almost all the requirements in the description. However there is one thing missing, the number of boats is not listed for each member in the compact member list.

There no input validation on the social security number, anything can be entered. It wasn't listed as a specific requirement but it should probably be done anyways. The project was a standard visual studio project and it was easy to get running.

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

Nothing fundamental but we did found some smaller issues.

1. The number of boats is not listed for each member in the compact member list, which was stated in the requirements.
2. There is no input validation anywhere, a new member can be added with an empty name and personal number. There wasn't a specific requirement for this but it should probably have been done anyways.
3. It's possible to select other values for the boat type than the listed, then it will be shown as a number in the member list instead. This is related to input validation of course.

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?

No problem, everything worked very well.

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 and sequence diagrams are auto generated and seems to conform very well with the code. The class diagram does however not have any relation, those are in a separate diagram and we think there are some problems:

1. The class Application holds an instance to a Menu object and should therefore have an association to the Menu class, this is currently shown as a dependency which is weaker [1, Chp. 16.11].
2. There is no association between Member and Boat even though the Member class holds a list of Boat objects.

3. The association between Registration and Boat should probably be a dependency instead of an association because of no direct connection between these classes (member variables).
4. In the diagram the Boat class have a association to the RegistrationDAL which seems incorrect since the Boat class does not have any dependency on the RegistrationDAL class.

Is the Architecture ok?

Yes it's very good, the model view separation is clear. The only thing we found which we are a bit unsure about is the error message which is printed from the Program class which is not a part of model, view or controller.

But overall, there is very good model view separation, the model does not have any knowledge of the view. The user interface has a dependency on the model, which is allowed.

Is the requirement of a unique member id correctly done?

Yes it has been correctly done.

What is the quality of the implementation/source code?

It's pretty good, the naming convention used seems to be a bit inconsistent in certain areas. Some functions have names starting with a capital letter, some have not.

Also some member variables start with a underline while other do not. The guidelines of this are a bit unclear but it would have been nice if at least the same convention was used in the entire application.

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

It's a very good object oriented design, we don't have very much to say about this. Some information could probably have been encapsulated better, like the unique member id, which we should probably not be allowed to change id for a member object after it has been set.

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

Yes they are pretty clear, the class diagram is very detailed so that would of course be of great help. The diagram showing the relationships between classes contained several errors so that would be a bit confusing to us.

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

- The object oriented design which is very well done.

- Diagrams, apart from the problems already mentioned.

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

- The class diagram showing relations, since its wrong.
- Inconsistent naming in source code, it will look more professional.

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

Yes absolutely, but we do think the class diagram showing relations should be updated.

References:

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