Daniel Blixt
Daniel Nilsson
2015-10-08

Peer Review - Workshop 2

For: Sing Trinh

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

Nothing fundamental but we did found some smaller issues.

- There isn't any hint of how the personal number should be submitted so we had to guess at couple of times before we found the proper format.
- It isn't possible to change the boat type when editing a boat.
- The error message when submitting an invalid member id doesn't match the terms in the corresponding if statement in source code. The message says the id needs to be 1 or greater. But in the code it could be between 1 and 9999.
- You can't add two boats of same model and size, no reason to have this rule. Let say that one owns several kayaks. Kayaks length in general is the same so should be possible to add.

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 everything works as instructed.

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 diagrams are well drawn and corresponds well to the implementations. But the inheritance arrows from the different view classes that inherit from BaseView have the wrong direction [1, Chp. 16.1, 16.10].
However we think that they overall lack some detailed information. Having the most fundamental functions of the classes in the diagram would make things easier to implement. As it is now its is more like a Domain Model then a Class diagram (Larman visualieses the differences in figure 16.2) [1, Chp. 16.2]

Is the Architecture ok?Is there a model view separation?

Yes there is, the only thing we found is that there is a Console.print in the member class, more precisely in the DeleteBoat method when an exception is thrown. This should be caught in the controller class and printed in some of the view classes.

Is the model coupled to the user interface?

No, every interaction between the UI and the models is done via the controller. The model has no knowledge of the view classes.

Is the model specialized for a certain kind of IU (for example returning formatted strings to be printed)

No it's not.

Daniel Blixt
Daniel Nilsson
2015-10-08

Are there domain rules in the UI?
Not that we could find.

Is the requirement for unique member id done correctly?

No, the id have to be submitted manually so no automatic id is created.

What is the quality of the implementation/source code?
Really good, it was very easy to understand the code and what is happening where. The naming of the different classes and functions follow the C# coding convections [2]. No duplications or dead code was found in the source code. However as mentioned before about the DeleteBoat function we have some redundant code. As the implementation is now the program goes through the list to check if the boat to be deleted is in the list and then a remove(object) is called on the list. In fact it would be enough just to call remove since it returns false if the boat isn't in the list.

What is the quality of the design?
The member id could be more encapsulated, maybe set in the constructor and remove setter so when an id has been given it is unchangeable. As it is now its possible to change it later on.

As a developer would the diagrams help you and why/why not?
Yes they would in the phase of creating the needed classes. However when we would get to the point when implementing functions for commutation and interaction between our classes almost no information is given. It would be completely up to the developer to solve this. The sequence diagrams would be to a lot of use. They show exactly what to be done. Here we have the functions that we think could be in the class diagram. One small error though, the “DoWork()” in the beginning of the sequence diagrams has another name in the source code.

What are the strong points of the design/implementation, what do you think is really good and why?
Well designed, easy to understand and “follow” the code. Good naming. Really good looking UI and easy to navigate through the menus.

What are the weaknesses of the design/implementation, what do you think should be changed and why?
Automatic assignment of unique ID is not implemented. The class diagram is a little too simple, it should include at least some operations and attributes.

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

Daniel Blixt 2015-10-08
Daniel Nilsson

References

1. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062
2. https://msdn.microsoft.com/en-us/library/ff926074.aspx

Daniel Blixt
Daniel Nilsson

2015-10-08