Jovydas Urbanavicius, Jean-Pierre Salum, Mantas Remeika | 1DV607 | October 12, 2016

Peer review

Leif Karlsson

Clear instructions for running the application was provided. It proves that there were no problem running application.

When registering new member application does not require any format when entering personal ID number. It accepts any type of input.

Moreover, it accepts negative number as boat length. While trying to change boat length and entering string, boat length changes to zero. It should show error and leave boat old length.

**Architecture:**

* **Is there a model view separation?**

It is clear separation between model and view.

* **Is the model coupled to the user interface?**

Models are not coupled to view.

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

In our opinion the model used in this application is not reusable, since it was created specifically for this assignment.

**Is the requirement of a unique member id correctly done?**

Implementation of unique member id in this application considerably satisfies requirement in the assignment.

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

* **Code Standards**

Classes and methods in this application do not have any documentation which could help other developers understand what is the purpose of each class and method. Documentation would also help for developer of this program to remember purpose of each unit in the application.

* **Naming**

Code is written according to all Oracle Naming Conventions [1].

* **Duplication**

We couldn’t find any code duplication.

* **Dead Code**

We couldn’t find dead code in the application code.

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

In this code objects are connected using associations and not keys/ids.

Controller class handles all system events, which according to Larman is a feature using GRASP pattern [2, p.431]. Classes have high cohesion and low coupling.

We found hidden dependencies in member class.

Encapsulation is done properly using get/set methods and using private variables and methods avoiding global.

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

The sequence diagram and class diagram would give us basic information before starting implementation

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

The structure in this application is clearly made using Model View Controller pattern, which gives better organization and allows easier maintenance. Variables and methods has good naming so it is easy to understand.

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

Apart from few warnings mentioned above. We think this implementation is satisfactory and we have no further suggestions

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

Our team thinks that this application passes requirements for the grade 2.

# References

* 1. <http://www.oracle.com/technetwork/java/codeconventions-135099.html>
  2. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062