Workshop 2 - Design

Peer Review of Sebastian Svensson (ss222tb)

Architecture & design

Every class, model, view and controller, handle the functions they are suppose to handle. The view only handles the output to the console, the controller only handles requests from the models to show it to the view and vice verca. I still would have seen the input handled by the view, because it's happening in the view, and that it's then is verified by the controller.

It also might have been a good idea to create another controller which handles the actual data storage and focus the other controller on interaktion made by the user.

The design of the applications is both good and bad. No real clutter in the design, but the verbose list is pretty hard to read.

I also have an input on the save method. Won't the user expect that the data is saved when they have created a new member or added a boat. And what if a user want to save without quitting the application?

UML diagrams

Class diagrams contains too many operations to be efficient to read (p 250). Though, they still have good naming, it's really just to many of them.

Good separation between the model classes and the view.

Smart to separate the sequence diagram in different areas but it's hard to figure out what classes that are models, views or controllers without looking at the class diagram.What are the strong points of the model?

The strongest point of the model is that it’s clean and it uses good names for the conceptual classes.

Code quality

Personally, I would like to se a better naming of the created object classes so that they tell me if they are a model, view or controller. Example instead of naming the object class "reg it would be better to call it regModel or m_reg.

The method Start in the controller almost seems to handle the whole application by it selves. It’s way to long and would be better of divide each case into smaller methods with a good naming of what it does. In that way the naming could be selfexplanatory and the comments wouldn't be necessary. It would help the understanding of the code better and make it cleaner. What I want to see is that the view should send the users input to the controller, which then check with with the associated model class if the input has anything to do with a specific item.

View: "Hey, controller, the user just told me he wants item 4 deleted."
Controller: "Hmm, having checked his credentials, he is allowed to do that... Hey, model, I want you to get item 4 and do whatever you do to delete it."
Model: "Item 4... got it. It's deleted. Back to you, Controller."
Controller: "Here, I'll collect the new set of data. Back to you, view."
View: "Cool, I'll show the new set to the user now.”

The other classes are nice and simple and does only focus to fulfill their duties.

As a developer would the diagrams help you?

Unfortunately, I would be a little overwhelmed by the class diagram due to all the operations but the sequence diagrams are really good with no clutter.

What are the strong points?

The strongest point is the separation between the models and the views. The code quality is still good, but could have been worked with a little extra to make it even better.

What are the weaknesses?

The weakness is the class diagram with it's overwhelming list of operations. It would have been good if it only showed the CRUD-operations. But operations such as GetBoatID or GetMemberName isn't really necessary to show.

Passed the grade 2 criteria?

I guess, but I’m a little concerned with the class diagram and the method Start() in the controller.