Peer Review Workshop 2

For the domain model made by Joakim Holmevi (jh222qr)

Daniel Vedin (dv222bk)
2015-10-08

Application

Repro lacked an easy to find .exe to start. I had to download everything and start the program using visual studio (or start the .exe from the debug folder). The .exe file can however not be started when moved to another folder, I would have liked to see instructions on what is needed to get it running somewhere else.

Application worked excellent and did not crash once. Application also seems to include everything needed for the assignment.

Class Diagram

The class diagram seems to be missing the “DataController” class in the controller part of the diagram. Only “UserController” is present there.

UserController only uses dependencies, but according to the code, I think it is, for example, connected to MemberList with an association and not a dependency. The reason being that the UserController class has references to a MemberList instance. The same is true for the connection to ConsoleView [1, p252].

UserController should also have a dependency arrow to Boat, since it sends variables of the Boat type [1, p261]

The Boat class, according to the class diagram, is using an enumeration “BoatType”. The Enumeration does not exist in Boat’s code however, the Boat class is actually saving boatType as a string. The Enumeration exists in the ConsoleView class, but according to the class diagram, ConsoleView is not connected to the Enumeration. Very confusing.

Sequence Diagram

Only one sequence diagram exists, for input. The sequence diagram for output is missing.

C.User is misnamed, its actual name is “C.UserController”.

V.DataController is missing from the sequence diagram, just as in the class diagram.

:Program sends a “StartApp(view)” to C.User. This does not happen in the actual code however. In reality however, after the :Program creates a V.ConsoleView object and sends the view to it. Then the :Program sends a doControl(view) to C.User. This means the sequence diagram misses the creation of V.ConsoleView and also the calls to the two classes are misnamed.

I will not further comment on the sequence diagram, since I think it’s pretty obviously created before the actual code was written. This means the diagram does not conform to actual code.

Architecture

Code is divided according to the MVC pattern.

The models are not coupled to the UI in anyway. This means that the UI can be changed without changing the models, which is good. The models do not return formatted strings either, which is also good.

The UI seems to not have any domain rules, which is also good.

The only thing I can comment on here is that UserController prints to the console when it catches an exception. This should be the views job, as it breaks the MVC otherwise.

Unique member ID

The code creates a unique member ID for each member and it works well.

Code Quality

The code follows traditional code indentation standards and is easy to read and understand.

Good names are used for methods and variables.

A quick look on the code did not reveal any code duplication.

The MemberID in the boat class is unused. It's not referenced anywhere else in the code. The GetBoatList method in the UserController class is also not used anywhere.

Design Quality

Code is connected with associations and not IDs.

Code and design follows GRASP, as each class has a responsibility. Other examples where the code follows GRASP is that the view does not care if the user input is correct, it lets the controller figure that out instead [1, p312].

One could argue that ConsoleView has low cohesion since it's so big and does so many things. I think the class is OK though, it just happens that its responsibility in the application is big. Even though the class is big and bulky, it only sticks to its one purpose: to output things to the console. Here, one could argue that outputting a specific member and its boats and outputting a menu is different (and it is), but I think it is not completely wrong to have both in the same class. At the same time I do not think its wrong to have them in different files either. It does make the class more bulky and hard to maintain however, so this could be something to think about in the future. [1, p314-315]

Overall I think that the application has low coupling. All dependencies and associations makes sense and is necessary. Obviously, it is not weird that the view needs to know what it outputs and it is not weird that the controller knows what it is controlling. Similarly, it is not weird that the controller has a dependency to "Member" when it has an association to "MemberList". One could be forgiven to think that a MemberList would always contain Members. This however could present problems in the future for other kinds of application and perhaps one should try to create a controller that only needs the MemberList association without knowledge of what an actual Member is. For this assignment however, I think it's OK. [1, p299-300]

The application uses several static variables. In the DataController class, both "path" and "file" are static and In the ConsoleView class both "clist" and "vlist" are static. While I can understand why the "path" and "file" in DataController are static, I think the "clist" and "vlist" ones are unnecessary and could have been avoided.

I could not find any hidden dependencies in the code, which is good.

Information seems to be encapsulated.

The code is defiantly inspired by the domain model. The classes uses the same names that could probably be found in the domain model as well, such as "Boat" and "Member".

Overall

The code itself is very well done and works excellent. However, the diagrams needs some real work.

Strong points:

- Code quality
- Architecture

Weak points:

- Diagrams
- UserController outputting to the console

Because of the problems with the diagrams, I do not think you have passed grade 2 for this workshop. Redo them and do them right. When you do, you will eat grade 2 for breakfast. Do not forget to fix the Usercontroller class outputting to the console as well.

References:

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