Peer Review 1 of mc222nk

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

- No input handling(which wasn't a requirement), but you should not be able to write letters as a personal number.
- Hard to find bugs since the interface was not running correctly.

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?

- The interface was unrunnable, Nothing in the Interface ran without problems and some commands didn't work at all. Maybe not an updated version or something else went wrong when uploading it?
- We can create a member and show it in compact list, working
- We try to add another member and it replaces the old one.
- We try to delete a member by its ID and it doesn't work, nothing happens and the compact list still shows the member.
- Get member does not work.
- There is no Verbose list.
- Update command does not work, same nothing happens.
- Boat create does not work, same nothing happens and no show in the compact list.
- Boat update does not work, nothing happens.
- Boat delete does not work, nothing happens.

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 delete boat diagram shows a method in member class called DeleteBoat(boat) but it is called RemoveBoat(boat).
- Boat update diagram “UpdateBoat(SocialNumber...” should be UpdateBoat(memberId…
- ”member->GetMemberId()” is missing from many diagrams, it is used in some methods and is calling to the member class so it should be in the diagrams.
- GetMemberList method has many “get” methods which it does not show in the diagram.
- Member lookup diagram has a method “GetMember(memberId, name, socialNumber, newMemberId)” but in the code it is only “GetMember($memberId)”.

- Member update diagram there is “Update user(...)” should just be Update. And in “UpdateMember(memberId,name,socialNumber,newMemberId)” the “newMemberId” is written as “newId” in the code.
- The class diagram is missing some methods and attribute. “MemberRegistry” class is missing “GetNewMemberId()” and “FindMember()”, the attributes “members” and “pdo”. “Member” class is missing the attribute “boats”
- Html files should be included in the class diagram since it is part of the interface. And right now the name “html” is not telling very much about what is in there except for html files.[1]
- Associations in the class diagram are weird and not helping.[2]
- The class diagram should show dependencies[3]

Is the Architecture ok?

- Yes it is, but html files could be put in the view package since it is part of the interface.
- There is a good model view separation

Is the requirement of a unique member id correctly done?

- It is not correctly implemented, since there is no indication that a members cannot have the same id. The code shows that a member can be updated to any Id no matter of the existing ones.

What is the quality of the implementation/source code?

- It is clean, good comments and easy to follow even tho we are not php experienced.
- Good names.
- Good separation of the classes, making them independent as much as possible.(not object calls back and forth).
- No duplicates or deadcode
- What is negative is that there is no input handling.

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

- It is good object oriented. There are no calls being made to attribute in other classes without an object attached, it is done only through object calls using getters and setters.
- The design is great and gives a very clean look of the code but the boat types could be individual classes extending an abstract class called boat for a more object oriented design.

---

[1] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 13.6. Guideline: Design with Layers

[2] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 16.4 Ways to show UML Attributes: Attribute Text and Association Lines

[3] Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062, Chapter 16.11 Dependency

- Keeping the database and other code seperate made the code much easier to follow and understand. Instead of just mashing them together in every method.

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

- Yes they would because diagrams were easy to read and to follow through the code.

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

- The architecture, comments and naming is good. Makes non messy code.
- Diagram were very easy to follow.

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

- The user interface is not working.
- No input handling whatsoever.

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

- With some fixes to the interface and functionality then we think it should pass. It looks like it is not completely done but the things that is done looks good.
- Functionality being that a member should not replace the old one when a new one is created and a member id cannot be updated to an existing member's id and so on. But if this has with that the interface is not working, we could not tell. The code it self seem to lack handling of user inputs.