Peer review: Abdirahman Duale

Compile and Use:

There were no instruction included in the GitHub repository explaining how we should run it. The files included were not in a project, this is not necessary but it make things easier for the tester if it is. It was very unclear on which class were the "main-class" so it was confusing on which class the system should be started from.

Assuming "MainBoat" was the "MainClass" we were not able to run the code as the system were asking for a "Boat file" which was not included. If we however deleted some commented code we could select which path the "Boat file" should be in, though what "Boat file" actually is is still unknown.

The system seem to require specific files and this should be clearly stated in e.g. a "readme".

Does the implementation and diagrams conform?

There were no included diagrams.

Is the architecture ok?

Is there a model view separation?
No. As all the classes where in the same package. No other separations were defined.

"Separation of Concerns, even if not perfectly possible, is yet the only available technique for effective ordering of ones thoughts, that I know of " - Edsger W.Dijikstra.

Is the model coupled to the user interface?
Is the model specialized for a certain kind of IU (for example returning formated strings to be printed)
Are there domain rules in the UI?

We are not able to answer these questions because we were not able to run the code.

Is the requirement of a unique member id correctly done?

Yes. Using nanotime for a unique ID may not be the best solution since IDs could technically be the same (although very small chance). We would like to see a method that checks whether the new generated ID is already used.

What is the quality of the implementation/source code?

Code Standards:

The code is not indented properly and there are multiple empty lines. Otherwise we think it's good.

Naming:

The methods have good names as we can clearly tell what they are supposed to do by the name alone.

Duplication:

MemberFetcher and BoatFetcher are duplicate code and could be done using a single method/Class with an if-statement included.

Dead code:

There are multiple methods that are never used e.g. setName, setPersonalNumber, setUniqueId to name a few.

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

We think that it is too many classes e.g. BoatFetcher, MemberFetcher and memberIdGenerator could have been methods.

As far as we can tell the code seems fairly object oriented.

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

Good methodnames that make the code easy to follow. All necessary methods are present.

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

Model/View is missing. Too many classes. No instructions on how to run the program. Hardcoded paths in the program. No diagrams included.

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

Hard to tell because we were unable to run and test the program. Judging only from the code and the fact that there is no diagrams it's not a passing grade.