Peer review

Reviewer: Max Decroy Strandlund ms224cq Reviewed: pe222gq

Compiling and running application

The applicaton runs perfectly. I did not find any bugs.

UML diagrams

The diagrams seems fine and represents the application correctly.

The class diagram could have had some central methods and variables to make it easier to use them as an entry point for understanding the code, as seen in Larman [1, chapter 16].

Architecture

The architecture seems good. There is a clear separation between model and view and the model is not dependent on the view in any way I could find.

According to the specification, every user should have an unique member number. This is lacking in the implementation. This should be added, and maybe used instead of the social security number as a database access key?

Implementations

Naming is great, toghether with extensive and clear comments they make the code easy to read and understand.

Coupling between classes does not always follow standards. For example, in the *MenuOptions.java*, line 12, `new Registry()` is called. This means that if another coder would want to change to a different Registry class, they have to edit the source code. Best practice for this might instead be to create the registry in *main.java*, and use it as a parameter for the `MenuOptions()`-class. Same goes for the creation of `Database()` within the `Registry()`-class.

Error handling is done by throwing exceptions in most places (ex. the Member class), and if-statements and console outputs in others (ex. MenuOptions, line 133). This is fine and the code is understandable, but it is inconsistent.

The `Database()`-class uses strings as commands. For example, a member is deleted form the databaes by doing `Database.GetMemberQuery("delete", member)`. Maybe these should instead be separate functions, to make it easier to use the Database class? As example, the member could be deleted by doing Database.DeleteMember(member). GetMemberQuery() could be a private function within the Database class, used by Database.DeleteMember(member).

Design

The design is great.

There could be lower coupling between the Registry and Database, as mentioned above. As it is now, the Database is not as encapsulated as it could be. Maybe one could even add an inteface between Registry and Database? Se Larman [1, p. 435].

Conclusion

The code follows standards, the classes have clearly separeate and well defined.

The design and implmementation should surely pass the crietrias. I think adding some more information to the class diagram and fix the missing member number might be the only changes really important.

References

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