Deliverable #4: Code Review Strategy

Review your assigned code on the following:

1. Readability & Maintainability: Are each of the functions/methods documented, with details on the function/method's purpose and parameter type/values? Is all the code covered by comments? Are variables appropriately named?
2. Code Design: Are the classes following appropriate design patterns? If not, how can the code be re-designed?
3. Functionality: Is the code's purpose understandable (are you able to follow through the code)? Is there any unnecessary code? Are exceptions/errors handled correctly?
4. Testing: Are there automated tests for the code? Is each test's purpose clear? Are all cases covered?

Deliverable #4: Code Review Summary

Assigned Code:

- Jeremy: accounts package, DatabaseIO
- Violet: main package, passwords package, test.account package
- Kyle: gui package
- Bilal: database package (except for DatabaseIO)

Jeremy: Basic Account structure established via User Class enables the idea of extendability. While the classes have yet to be fully implemented, it is structured in a way where adding the code and behaviours won't be difficult. Therefore there is not much missing in terms of commenting or purpose of code. One thing to note is to find a mechanism to keep track of what the different subclasses of User can or can't do for actions so when generating the GUI for each account, we know exactly what to display. The only issue now is properly linking the 3 components of User, GUI and Database since a User utilizes database manipulation methods, but a GUI needs to display that/provide a workspace to be shown. To handle what User type can do what, a suggestion of using an interface could be an optimal choice in providing behaviours for specific types. Tests for DatabaseIO need to be automated or clearly explained in written comments. Tests for the accounts are automated.

Violet: For main, the structure of it is clear and easy to understand due to good naming. However, some javadoc and comments should still be used to make it more easier to follow as new features could be added to it in the future, which might make the code longer and more complex. DatabaseConnector is relatively small, as it has no more than 5 lines, so the only thing needed is a Javadoc to make it looks more professional. For passwords, it also lacks javadoc and comments. But the structure of it is good, which makes it could be test and changed easily. For userTest, as in all the test cases, it is needed to create a same user, it might be better to put that under before, in which way the code would looks more neat. Other parts are good as the naming is good and it tests all the cases that needs to be test.

Kyle: The gui seems to be very straight forward. There is currently very little GUI component to test but of the components present, the login is fully working. The code for each panel is separated into different classes. The use of comments are very basic, but because the GUI is mostly pre-implemented methods, the comments added need only state the desired outcome, meaning the documentation is not as important as the result. The maintainability of the code is very good, because all panels are segregated. All poor maintainability standards are inherent in the pre-implemented swing elements and cannot be avoided. Although I expected more code to have been present, the quality of the present code is of high standards. With regard to the intended design, the return buttons are missing. Testing for the GUI has to be detailed in written instructions.

Bilal: The database methods are lacking documentation, thus it is relatively difficult to understand each parameter's purpose and further help is needed to properly utilize these methods. However, it is fairly easy to understand each method's purpose due to the good method naming used. Additionally, there is a substantial lack of commenting as well, thus making it difficult to follow through the code and interpret how the method is achieving its purpose. This problem is mitigated by the use of good variable naming. The database methods are very well-designed and are ready for usage by the middle-end and front-end (GUI) to manipulate database data either directly or via temporary storage in JTables. The missing documentation and commenting should be added retroactively, to simplify the process of utilizing these methods in the completion of the remaining User Stories' GUI. We need to attempt to automate testing for the database methods or write out clear instructions for manual testing of the database manipulation.

Deliverable #4: Code Review Debriefing Meeting

The video of our Code Review Debriefing Meeting can be found at this link (file was too large for git): https://drive.google.com/drive/folders/1En0c8miqlZGAE9qKnOns6PBM8n2dISTC