Refactoring Document

1. **ScoreCounter**
    a. The “counter” parameter of the ScoreCounter class was changed from an integer to a SimpleIntegerProperty to be displayed in the ScoreChart which meant that the add and subtract methods needed to be refactored to work with the SimpleIntegerProperty.
    b. The getCounter method also had to be refactored to directly take the current score from the database by using the getScore method from the DB.java class

2. **Login**
    a. Before refectoring, when registering a new user, the ID section can be blanked, but this is not a common thing to happen, so I added a new exception handler about ID should be more than 0 characters. So that the ID cannot be blank. Also, I dived the exception handler, for example, divided it like this:
    1. "ID must be at least 1 character"
    2. "Password must be at least 10 characters long"
    3. "User name already exist"

    To give users a more clear understanding of what the user is missing.
    b. When registering with the same ID that exists in the database, it won’t show any exceptions such as “User name already exists”, and it just moves on to the main login page. But I refectored this design issue by adding a line that checks the database ID and user input ID. If a user input ID exists on the database ID, then it won’t move to the main login page and it will throw an exception about “User name already exists.
    c. I refactored the database to prevent the code smell called “code duplication”. Before, in the database, database initializing code was duplicated in every method, but I made an init() method to prevent the code duplication.

3. **App**
    a. Reasoning: Initially, all application logic was within the App.java file, including the main method responsible for launching the application. However, as the application became larger, it became necessary to separate concerns and improve code organisation.
    b. Solution: To maintain a clear separation of concerns, a new Application.java file was created to house the main method responsible for launching the application. Each class now has a single responsibility, making the codebase more modular, maintainable, and easier to understand. Additionally, separating the entry point of the application into its own file improves readability and scalability, especially in larger codebases. The application.java now handles the login and registration logic and the app.java handles the database interactions and displaying the application windows.