## Improvements from phase 1

- For the FilesReaderWriter class, we use one method which save object into the ser files and cast the different use case into object in DemoManager rather than one method for one use case in FRW

- Also, for the FilesReaderWriter class, we let DemoManager create new empty use case if cannot read the use case from file(which means the file is empty) rather than let FRW create new use case

- We got rid of all the dependency on scanner as well as the high coupling between presenters and controllers by using java swing instead of Text UI

- We obtained better adherence to clean architecture + decoupling + encapsulation by getting rid of all method calls to entities in controllers so that controllers only call use case methods

- In phase 1, we have two distinct classes, AdminUser and User. However, User contains all of AdminUser's methods and more, so we think it would be a good idea to make AdminUser the parent class and User the child class, which is better for open-closed principle (ex. It'll be easy if we need to add other kinds of User later on) and avoid duplication.

- In phase 1, we only had two presenters, DisplaySystem and System Message. However, we found that DisplaySystem doesn't really adhere well to the Single Responsibility Principle and most of the things our presenter prints in phase 2 are system messages. Therefore, we deleted the DisplaySystem and kept the SystemMessage class. And then we found that printing actions require a specific type of formatting, and thus we added these two new presenters: AdminUserActionMessage and RegularUserActionMessage.

- Create account: now we use regex to validate an email

- We also use regex to check if a string is an int. By doing so, we don't need to try and catch number format exceptions.

- Solved hard code issues

- Use case classes created methods to avoid controller call entities directly.