Observer Design Pattern for undo functionality (Edward)

When I was implementing the undo functionality, I used the Observer design pattern.

The basic mechanism of undo is that the system will record every modification or any action that the RegularUser class has made. Thus, I separate the cause and effect of undo functionality.

Cause: The user has made an action
Effect: A piece of history is recorded so later on, this action can be undone.

To separate them, I made the RegularUser to be the Observable, and the RegularUserActionStack to be the Observer. Once an action is detected from the user, the execute method in the user will be called and then the RegularUserActionStack will call the update method. Thereafter, a piece of history is saved.

Dependency injection for VacationModeActivity (Locky)

When I was implementing the VacationModeActivity. I used the Dependency injection for regularUser by injecting the RegularUser from MainActivity instead of just instantiating a RegularUser Object in this program.

This is really helpful for decoupling the program and reducing code smell.

Singleton Design Pattern for demo functionality (Yuzhan Gao)

Singleton design pattern was used for implementing the demo functionality. Based on the phase two instructions given to us, we need to create a new user type that is specifically used for the demo feature, which is DemoUser. Obviously, we only need one and only one universal DemoUser instance. So I applied the singleton design pattern to make sure that the DemoUser class has only one instance and every time we run a demo, the same demo user will be used.

To achieve that, instead of using a file to save user information like what we did for regular and admin users, we just initialize the same demo user in the UserData class.

Dependency injection Design Pattern for smartSuggestion method in ItemSuggestion class (Chen Hua)

In the smartSuggestion method, I used the Dependency Injection design pattern.

- **Why**: I used this Design Pattern because the class can benefit from decoupling between CleanText class and ItemSuggestion class.
- **How**: I implemented Dependency Injection by injecting an instance of CleanText object into the smartSuggestion instead of creating an instance of CleanText inside the smartSuggestion method.

Comparator used in UserInfoPresenter and IntegerComparator (Locky)

In UserInfoPresenter (A class designed as a presenter to show the favorite partner and currently traded items), I use Comparator. ,
Why: A Comparator can help reduce the code smell and better encapsulate the algorithm. In UserInfoPresenter, I need to check each partner’s so as to decide which one should be the “favorite” by sorting the partners.
How: IntegerComparator is defined so as to compare the times of each partner’s number of transactions. Then in the UserInfoPresenter, we new an IntegerComparator in the sort() to find out the most favorite partner.