Thomas Hart
CS 340
Milestone 2C Report

Observer Pattern

I'm using the observer pattern between my PostStatusHandler and PostStatusObserver classes to post a status. The PostStatusHandler class is the subject, which holds the PostStatusObserver (the observer). When the MainActivity View is changed, it notifies MainPresenter, which performs the postStatus operation. The PostStatusHandler is aware of the state of that operation, whether it succeeded or failed, so it notifies the PostStatusObserver. The PostStatusObserver relays that state back to the view. Both PostStatusHandler (in StatusService) and PostStatusObserver (in ServiceObserver) are technically in the model layer, if you count the service layer as part of the model layer.

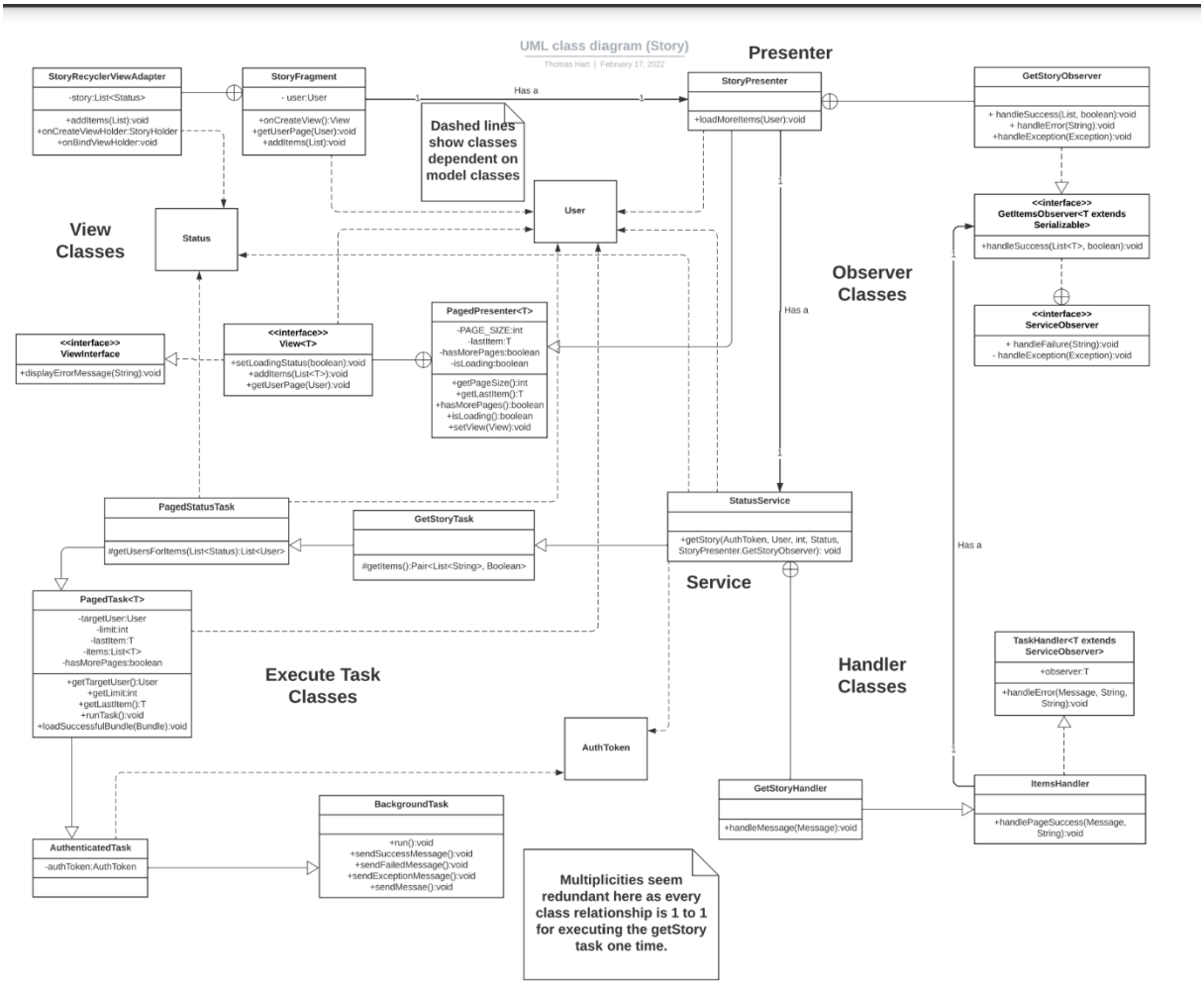
Generics

I used generics in my TaskHandler class because it is the base for the all the handlers. Generics in this situation make all the nested handlers have an observer of type <T extends ServiceObserver>. This way, any necessary observers can be passed to the handlers while eliminating code duplication.

Template Method Pattern

The template method pattern is used in each of my handler classes. We'll use the FeedHandler class to demonstrate. It extends ItemsHandler, which extends TaskHandler. TaskHandler and ItemsHandler use FeedHandler's template to call methods. If the operation fails, FeedHandler calls TaskHandler's method handleError to handle both failures and exceptions. If it succeeds, FeedHandler calls ItemHandler's handlePageSuccess method. FeedHandler therefore has the template for the operations which should be called, while TaskHandler and ItemsHandler execute the detailed methods. I set it up this way to remove code duplication. All the handlers for each operation do the same.

UML Class Diagram for Getting a Story



UML Sequence Diagram for Login