Final Design Document – Felix Li

Introduction

I did the Hydra project by myself. I implemented no special bonus features except the memory challenge.

Overview

My project uses the Model View Controller design pattern. As shown on the UML file, there is a TableModel class, which serves as the logic for the program, and a TableView class, which contains both Controller and View aspects of the MVC design pattern. To start a game, one must create a TableView object and call it's beginGame method, which will do various things including calling TableModel's distributeDeck method to initialize the players, their cards, and the hydras. It then calls the play method of TableModel, which performs the first move and begins to call the notfiyView method, which calls TableView's notify method. That is essentially the handing off responsibilities from the Model to the View: the View will accept input, make any changes to the game state (Model), and constantly print out the game visually all through the update function.

I implemented a Card class and a Joker subclass. They basically store the information of a card : value and suit. The pile class contains a vector of unique pointers to cards. It has methods to move cards between piles, and get information regarding the first card. The player class is basically a container for the player's information, including their piles of cards and their number. It is very much a part of the TableModel class, and is therefore rather coupled.

Design (describe the specific techniques you used to solve the various design challenges in the project)

To prevent Jokers from special casing, I made the joker a subclass with it's own getName function so that I would not have to check for that every time I wanted to print a Joker's name.

Resilience to Change (describe how your design supports the possibility of various changes to the program specification)

To increase resiliency, I used the MCV design pattern so that if there needs to be a new user interface, I could easily replace the TableView class with a new one that does the same thing except with a new implementation. The logic and the necessary functions maintaining the game data still resides in the TableModel class, as well as everything else that can be reused. In

addition, I made sure to encapsulate the Pile class so that it's implementation is hidden and if I changed anything, it would still be usable. The same goes with the Card class.

Things I changed

I updated my UML diagram to include everything. Before, it only included the pile and card related classes and nothing else. In addition, I also deviated from my original schedule. I did not schedule appropriately for the other exams, which made me have less time from doing this project.

Extra Credit Features (what you did, why they were challenging, how you solved them—if necessary)

I did the challenge where all the memory was handled automatically without new or delete statements. Firstly, I implemented all the pointers using unique pointers. Secondly, I made sure all variables and objects were stack initialized so I wouldn't have to delete them. Thirdly, I use STL:vector instead of an array so that the memory was managed automatically. Overall, it wasn't too bad because this also saves the hassle of tracking down memory leaks.

Final Questions (the last two questions in this document).

1. What lessons did this project teach you about developing software in teams? If you worked alone, what lessons did you learn about writing large programs?

The main thing I learned about writing large programs is that it takes an enormous amount of time, which I did not accommodate for completely. I believed it would only take a week, but in reality, it took a lot longer. I learned that it is very important to have a solid plan in place, otherwise, it will be very difficult to continue once the project becomes big.

2. What would you have done differently if you had the chance to start over?

If I were to start over, I would give myself at least another week to do this project to make it polished. As stated above, I was running short on time towards the end of the project because I underestimated the amount of time that it would take.