Final Design Document

Written by Bingxun Lu
Student ID: 20663297

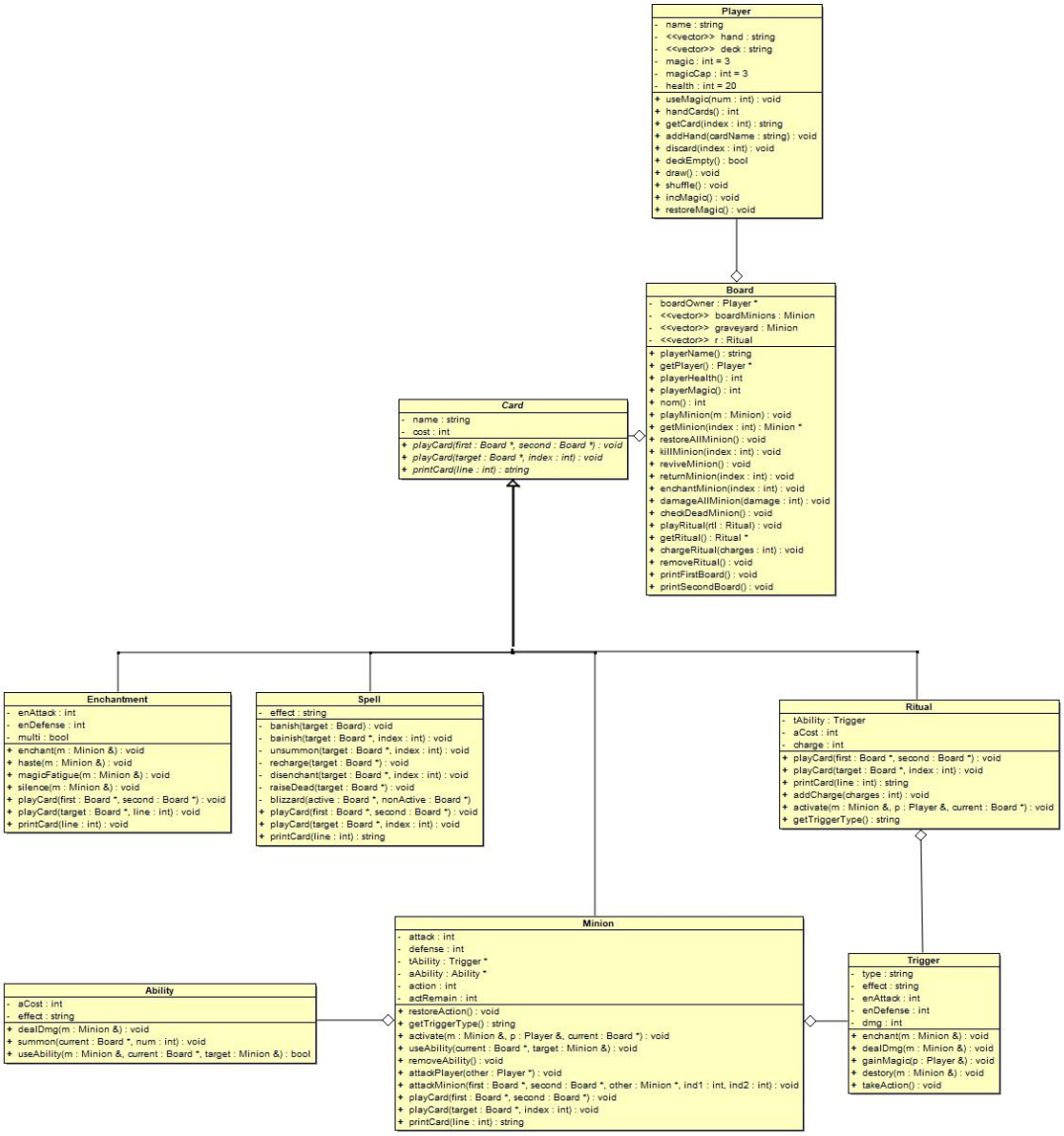
Introduction:

I chose Sorcery as my final project and I worked alone on this assignment. There are a few features of the game that I did not successfully achieve: Graphics mode, the card Silence, the command Inspect, the minion leaves play trigger. There are also a few bugs that I have already discovered but did not have time or cannot fix: Under some unknown circumstance, minions with summon ability are not able to use abilities and not accessible but can still be printed on the board, minions created by the summon ability cannot properly take damage form spell Blizzard, the minion enters play trigger does not properly work with all the summoned minions but only the last one. Besides these features of the game, other functions work very well under my own test, but I cannot guarantee that they have absolutely no bugs. All the commands are expected to be valid and legitimate. I did write some error messages to be printed out when commands are not valid, but they may not cover all commands.

Overview:

I used 9 classes and a header file Card Library to structure my Sorcery game. The classes are: Board, Player, Card, Minion, Ritual, Spell, Enchantment, Trigger, Ability. Board has a pointer of Player and Cards located on the board, graveyard, and ritual. Minion, Ritual, Spell, Enchantment are four derived classes of Card, both Minion and Ritual have Trigger(passive ability), and Minion also has Ability. I created a Card Library because I want to initialize Cards through the library, specific details and reason why I did this will be introduced in the design section.

Updated UML:



(This UML might be hard to read because my UML editor does not support exporting the diagram as image of PDF file and I had to use screenshot.)

Design:

There are many challenges writing this game, the first one was when I tried to load a deck from the file “default.deck”. Loading cards form a deck means initialize all the Cards in the deck, but without knowing the type and details of a Card, it is impossible to create Card object using only the names in the file. Then I tried to maintain vectors of strings as a Player’s hand and deck. At the same time, I created a Card library file using map<string, Card *>. By doing this, I can initialize all the cards before the game starts, and use the card name to get the Card that I need form the map dictionary. Another benefit is that since I stored Card pointer in the library and Card is an abstract class, I can call the playCard() method on every object in the map.

The virtual method will decide which playCard() it needs to call and I do not have to worry the type of Card.

Another problem is how to print the Card, hand, and board. I first thought of MVC pattern, but I am very unfamiliar with the design pattern. Therefore, to avoid writing code that I am not familiar with, I decided to print in a special way. First, it is obvious that board and hand must be printed line by line, and all the cards have the same structure, so I thought of separating the printing task into lines. Second, I observed the structure of all four types of Card, and discovered that Spell and Enchantment do not change at all, Ritual's charge changes, and Minion can be modified in various ways. Then, to print Spell and Enchantment, the name of the Card is used to decide which Card is printed form the files "spells.txt" and "enchantments.txt". Ritual is printed in the same way besides updating the charge of a Ritual. In terms of minions, since there are so many elements that could change, I made a print structure for every line of a Minion. The print function is virtual, takes an int and returns a string, where the int is the line number of a card and string is the content on that line of card. Finally in the displayBoard and displayHand function, the printCard method is combined with some special layouts and printing now works perfectly.

Enchantment becomes very challenging when it can be removed from a Minion. I do not have time to complete this feature but I still have some idea about how to achieve this goal. My idea is to add a list of Enchantment as a new field of Minion. When a minion is enchanted, an Enchantment is pushed into the list but other fields of the Minion remains the same. When the Minion needs to be printed or interact with other objects, it must be "finalized" first, meaning apply all Enchantments stored in the list temporarily. I think it will be much easier to remove only one enchantment in this way, and it should also enables the command "Inspect". The difficulty is how to "finalize" a minion.

Resilience to Change:

I would consider my code is at a medium level of coupling and cohesion. All my methods are useful and pertinent. If some rule of the game has changed, it is very possible to just change the parameter or call another method to adopt the change. However, many of my methods depend on methods from another class and I have not thought of a better design to minimize the connections. It might be risky to change one part of my file while keeping other parts work normally.

Answers to Questions:

1. Question: How could you design activated abilities in your code to maximize code reuse?
Answer: I created an Ability class and used a pointer of Ability as one field of Minion. I can use the Ability class to build a collection of all abilities. For instance, if I need to enchant a minion to give it a new ability, I can search in the collection and apply it to the minion.

2. Question: What design pattern would be ideal for implementing enchantments? Why?
Answer: The ideal design pattern for implementing enchantments is Decorator Pattern because we are adding new features to minions at runtime. Also, since we want to inspect a minion latter to see all the enchantments, it is necessary to keep the pointers of all the enchantments applied to a minion, otherwise we can only know what a minion looks like in the end instead of how the minion was modified step by step.

3. Question: Suppose we found a solution to the space limitations of the current user interface and wanted to allow minions to have any number and combination of activated and triggered abilities. What design patterns might help us achieve this while maximizing code reuse?
Answer: Decorator Pattern

4. Question: How could you make supporting two (or more) interfaces at once easy while requiring minimal changes to the rest of the code?
Answer: We can use the Model-view-controller pattern. We create different views for each interface and use the controller to connect our source code and how the information should be displayed. Then there should be very minimal or even no change in our model.

Final Questions:

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?
Answer: As I worked alone on this project, I found that it is really important separate the whole project into parts and steps. When I was doing assignment 3 and assignment 4 (relatively bigger programs), I did not make any plan of attack beforehand. I just started coding in the .h and .cc files and compile to see the errors, then fix errors one by one. However, it usually turns out that there are way many more errors than I thought. Fixing those errors could be really slow because I have no idea which part of my code runs and which ones do not. As a result, I wasted so much time and finally got very bad marks on those two assignments.
In this project, the Advice section really helped me a lot. It gives a series of steps and each step is straightforward. I used it as my guide to code, and the result is remarkable: I completed the project in two weeks by myself. The reason is simple, by splitting the program into steps, it is easier to fix any errors. When you know which part of the code is certainly working, it is much faster to locate the error later. Therefore , no matter how large the program grows, it is impossible to go horribly wrong, just like a building with a solid base. In conclusion, the lesson I learned is: Take as much time to make a plan before writing any code, build the program from very basic steps.

2. What would you have done differently if you had the chance to start over?
Answer: I would spend more time to structure the classes. I am not very satisfied with my current structure as I mentioned in the Resilience to Change section. Also, many methods in Ability and Trigger should have been made private. But I am not very sure that no methods from other classes called them, therefore I decided not to take the risk to change my code as it is getting close to the deadline.