CS 246 Final Project
Straights

Project Design

Justin Li

Introduction

1. Project Setting

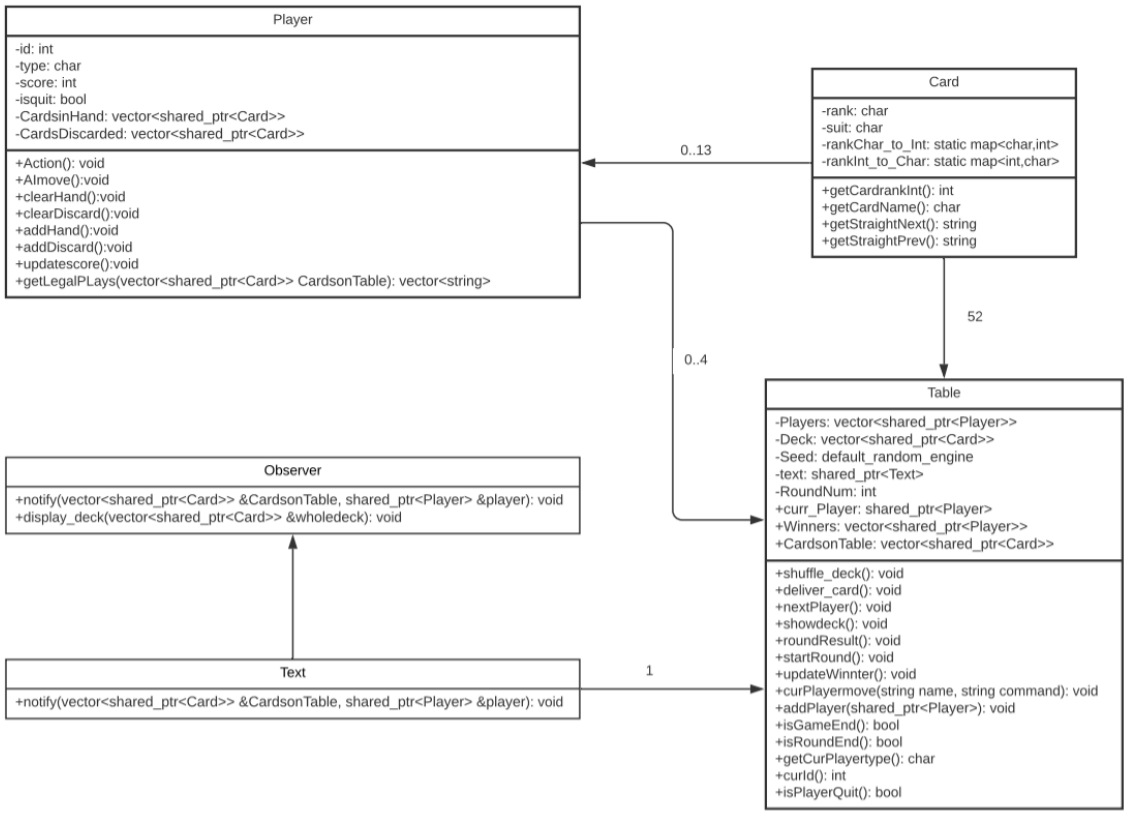
This project is called Straights, which is an interactive card game. This game needs a standard 52-card deck (https://en.wikipedia.org/wiki/Standard_52-card_deck) and usually 1 to 4 (human) players. The players are assigned with 1, 2, 3, 4 in order. At the beginning of each round, the 52 cards will be equally and randomly distributed to 4 players, then the first turn of the round will depend on the player that has the card 7S in hand.

2. Game Rule: A player can play a card in his hand which is also a legal play in his turn, if there is no legal play, the player must discard a card in his hand. A round ends when there is no card in all players' hand. When a round ends, each play will get the score of that round based on the ranks of their discarded cards, if there exists a play has score equal or greater than 80, the game is over and the play has the lowest score is the winner (can be more than one winner); otherwise, a new round will begin.

* Fun fact: the game can be played by setting 4 computer players

Overview

The project applied the MVC design method, the following is the UML diagram that can represent the design.



The module contains the class called **Player,** it stores the information of a typical player and provides some interactive method. Another class in the module called **Card**, it stores the information of a typical card in the game. The class **Observer** is the observer in design pattern, it has a derived class called **Text**, it displays the current game status and the table via text. The class **Table** is a crucial class in the design, it behaves as a controller in the design pattern and processes the commands read from users.

Design

The effective way of calculating legal plays for a player

Calculating a player's legal plays in this turn requires the cards on table the player's cards in hand. Since the rank of a card is a char type and it contains some letters and numbers, so it's hard to compare their rank. An effective way is to define two maps, one is the type <char,int>

Another one is <int,char>. These two maps can easily convert the rank between char and int. When calculating the legal plays, first step is to sort the cards on table by their rank, then separate them into 4 suits and use **Card::getStraightsNext()** and **Card::getStraightsPrev()** to find the legal plays in each suit. That's how the function getLegalPLays() returns the vector of string, the legal plays for that player.

Keeping all cards from deleting in the game

In this game, vectors are used to store the data of all cards, and once a round is completed, the vector CardsonTable will be cleared for the following round. Since these cards are stored in the field via shared pointers, the elements are removed once the vector of shared pointers is cleared. A wise way to keeping all these cards is to create a field called Deck in the Table class, Deck stores all 52 cards in the game, and it can also be used for storing the shuffled cards since in each round we shuffle the cards based on the previous round's cards order and the random seed.

Avoid getting memory leak

In most situation, it's hard to remember deleting object after newing it. This will lead to memory leak in the program. Then we need to avoid getting memory leaks and managing the memory. Then, handling all memory management smart pointers is the wise choice in this project.

Resilience to Change

When making some changes of the program, we can just modify the specific object and the class **Table**. Since this project is used MVC design pattern, if we want to change the interface, we can simply supplant the current view with various ones, like a graphical-based view, it doesn't need for any adjustment in the model, we can do a few alterations in the regulator. If we really want to change the game rules, we can simply change the algorithms to process the game, computer player's strategies and the method for calculating the scores. For instance, when we want to add extra elements in the game, we can modify the class Table by adding some new functions.

Answers to the Questions in Straights Project

Question 1: What sort of class design or design pattern should you use to structure your game classes so that changing the user interface from text-based to graphical, or changing the game rules, would have as little impact on the code as possible?

I used the MVC design pattern in this project, if we need to modify the interface, we can just replace the current view with different ones, like a graphical-based view, it does not require for any modification in the model, we can just do some modifications in the controller. If we need to change the game rules, we can just change the algorithms to determine whether the play is legal, computer player’s strategies and the way to calculate the scores. For example, when we need to add additional features in the class Table, we can just update the Observer, the Text class will automatically follow the update of the Observer.

Question 2: Consider that different types of computer players might also have differing play strategies, and that strategies might change as the game progresses i.e. dynamically during the play of the game. How would that affect your class structures?

I will add an additional field to the Player class, which called AI level. The level indicates how smart of the computer player would be. As a computer player has higher level, it will use the better strategies based on the current situation in the Table. According to this, I will update function AImove() and add the some strategies functions based on computer player's level. This does not affect other class structures, because it just considers as the new features, we can just add or do some modifications on the algorithms.

Question 3: How would your design change, if at all, if the two Jokers in a deck were added to the game as wildcards i.e. the player in possession of a Joker could choose it to take the place of any card in the game except the 7S?

I will add one more field on the Card class in my design, which indicates whether this Card is Joker. Then, I will change the algorithms to determine whether the play is legal, computer player's strategies and the way to calculate the scores based on the new Card "Joker". If a player has the Joker, his legal play will be calculated only based on the CardsonTable, when the player plays a card that is legal but not in his hand, the Joker will automatically be the card he wants to play and show up on the table. Since this can be considered as "changing rules" of the game and we are using MVC design pattern, so other classes will be automatically following the update I would make.

Final Questions

Question 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?

Since this project Straights is a single person project, so I worked alone. I learned that a good design could make the classes easier to implement and increase the coupling. In this project, I knew that writing large buggy programs is hard and writing large correct programs is much harder. Therefore, making a plan for a project is the first step of the start and a detailed design can make the implementation and debugging be much easier.

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

If I can start over, I will do the following:

1. Implementing a graphical interface

Clearly, the text-based interface is lack of beauty. Nowadays, people's needs of software is more than just the program. A beautiful interface can better help people use the software better and easier. Therefore, based on the original basis, I will add a graphical interface to help users to use the program easily.

2. Adding “Joker” card mode

Only using the standard 52-card deck in the game can make the game to be easier, but sometime may makes the players feel boring. Therefore, adding a mode that involves the wild cards “Joker” will provides more possible move and outcomes for all players, which can make

the game be more arithmetic hard, this can provide more fun for the users who enjoy the game.

3. Implementing difficulty level of computer player

In my project, the computer players will only play/discard the first available card they have. This may make other human players feel less challenge. Therefore, I will implement some distinct difficulty levels of computer players because this is a good choice to provide more fun and challenge for human players. This is an effective way to improve users' experience in the software design.

Conclusion

In conclusion, the final project has a great difference from the plan I have, but they do have the same main logic. The final version of the project uses the smart pointer widely, which avoids getting memory leaks. This program didn't have the exceptions handling because not much time is given for implementation but it's still a good practice for coding and understanding the concepts in OOP and the future study of computer science.