Final Project Design Document

Straights

Taebin Kim

20888822

CS 246

December 15th, 2021

Introduction

My program allows single or multiple users to play a game using standard play card, *Straights*.

There will always be 4 players participating the game and each player will be given 13 cards to start with. Whoever has spade of 7 gets to start the game by playing spade of 7 on the table. Each player takes a turn after one another. Once it is a player's turn, they can either play or discard one of their cards if specific player wishes to continue. Only cards that they can place on table must satisfy some conditions. First, any card with rank of 7 can be placed on table anytime. Second, a card with the same suit and adjacent rank to cards that have already been placed on the table. Make note that ace and king are not adjacent to each other. If one does not have any card that they can play on the table, they must choose to discard one of their cards. Also note that one must not choose to discard if they hold any cards that can be played on the table.

Described rules allow the players to continue a round. At the end of the round, each player will update their score by adding ranks of cards that they discarded in the most recent round played. Ace corresponds to 1, Jack corresponds to 11, Queen corresponds to 12, and King corresponds to 13. The whole game consists of several rounds and continues until one of the players total scores exceeds 80 after a round. When the game stops, a player with the smallest score will win the game.

However, it is all gamer's instinct to rage-quit or quit the whole game in the middle. Once a player decides to rage-quit, the game continues in the same manner except that his moves and plays will be made automatically by computer-player. Once a player decides to quit the whole game, meaning shutting down the game in the middle, we execute the program immediately.

Overview

As mentioned in introduction, we have 4 major components – players, card, deck, table. When it comes to implementing a game of *Straights*, I thought it would be more efficient and easier to manage each component with a new type of object, which I called "game". Hence, we have 5 major classes to implement this game.

First major part of my project's design structure is to manage relationships between card, deck, and the table. Intuitively, it is usually said that "the deck is made of cards." Based on lectures on UML and relationship of classes, I have concluded that this relationship fits best to composition since we could also say "deck owns cards." A deck, especially, contains 52 cards. Similarly, a relationship between the cards and a table also belongs to composition. However, unlike to the deck, cards on the table are placed, or can be placed while following specific rules.

Second major component of my project's structure is to manage players. It may sound easy to implement them since there are only 4 players. However, as mentioned in introduction, there can be human and computer players in the game. As both human and computer players share many commonalities, it has been natural to have an abstract class, (general) player. As implementing and constructing plan to attack this problem, I have realized that there is no huge difference between computer player and human player when it comes to playing the actual game. Hence, I chose to implement most of methods in abstract class, player, and implemented two specific type of players in a way that their only distinction is their type as it would affect the program when one chooses to rage-quit.

Third major part of my project's design structure is to construct a game class that can manage all those components of this program accordingly. Most of the components mentioned above are to become fields of game class so that the game class can access and use appropriate methods of each of its fields.

Design

Throughout this paragraph, I will follow the flow of overview section to explain my design choices in detail.

To accomplish first part of the challenges, I have made use of vector and pointer to Card objects. The reason why I chose vector is due to its convenience to manage each element with handful of implemented function that I can use of. In addition, I have decided to make use of pointers to Card classes to solve this problem. Though I could have used vector of objects directly, I have thought this would be easier to implement with. This portion of design structure focuses more on class relationship. In addition, each class related in this portion of challenge, Deck, Table and Card have their implementation related to key concepts taught in this course such as constructor, getters and setters, and input and output operators. Another key concept, encapsulation, has been well applied in solving this portion of the challenge. For example, suit and rank of Card should not be adjusted easily, hence it makes sense to keep them private – specifically, we may allow access from others, but not mutate the fields.

To achieve second goal of this challenge, my implementation uses a concept of abstract classes. It allowed me to manage two different types of players easier. Though it was not very close to examples and explanations done in the lecture regarding to abstract classes, the concept of gathering different classes into one gave me a significant inspiration while implementing and attacking this portion of the challenges.

To achieve third part of this challenge, I have been able to find a solid solution while carefully considering class relationships and inheritance. Attentive consideration on each relationship of each components became solid foundation on how I would attack this problem. Though coming up with solid solution to a problem before implementation yet, considering class relationships allowed me to have flexibility to adapt my initial structure design depending on barriers that I have faced.

Overall, I have made reference to solid number of concepts taught in this course, such as special class members, encapsulation, standard templated library, design pattern, relationships and inheritance.

Resilience to Change

As mentioned, I have broken this game into components, and I believe that I have broken the challenge down into possibly smallest components possible. That is, when any changes occur in program's specification, it would not be likely that those changes would affect logic of implementation of several class, meaning that only minimal number of classes' logic would be affected.

As mentioned, the game class currently deals with the most of logic related to running rounds and a whole game. Then, any changes related to the flow or logic of the game mostly will affect game classes only. For example, suppose we wish to build a game with more players. Then, the only changes required in implementation is to increase number of players by initializing more players needed. Other changes regarding this game then would be related to each component of this program. For example, if we add joker, it will mostly affect the implementation of card and have little effect on game or other classes.

In conclusion, as I am reflecting, the way that I have implemented my program is able to deal with changes relatively easily which guarantees resilience of my program.

Answer to Question

1. *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 would have to add decorator design pattern to our design.
- I can make class called "computerComponent" where both ComputerPlayer and ComputerStrategies, a new class, are children of. ComputerStrategies will be our decorator class in decorator design pattern and its children will be several strategies to decide move depending on the state of the game.

2. *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 could approach to add this feature to my program by making a card abstract class.
- My initial approach to break down this task was to think of joker as a blank card where its suits and ranks are yet decided.
- We could implement joker cards in a way that its suit and rank are determined when any player decides to play that card and place it on that table.
- Then, for joker class, we could have its suit and rank to be public or implement mutator for those fields so that either table or player can modify
- However, this flexibility is not what we want for other cards. Their suit and rank must be set and should not be changed. Then, we can implement normalCard classes so that each attribute is not public and do not provide any mutator.

3. *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? Explain how your classes fit this framework.*

- I can start with building a Subject and Observer class. We will modify our Game class to be our concrete subject.
- Each major components, then will be edited so that they become a concrete observer to the game.
- Then, when the game class decides to notify all the observes towards it, then each

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?*

Throughout this challenge, a lesson that I have learned the most in terms of writing a large program by myself is the importance of planning before the actual implementation. I have spent relatively less time in preparing my plans to attack this problem, comparing to other assignments that we had and the size of this project. Hence, I have had to start implementing without feeling very well prepared. As I was close to implementing each file and felt ready to compare, I was initially overwhelmed by the amount and number of errors that I had to fix. Hence, I have adapted my strategy as I was solving one problem at a time. However, one cons of this approach was that I have had to revisit previous problems and solve them again in different ways to solve next challenges. It was not pleasing experience of coding and, to be honest, I was quite emotionally stressed and overwhelmed.

Though I have learned that planning is important, I have also learned that the plan could always change as I make progress. However, I have learned that I should make minor changes first to make major changes. I have sometimes taken an approach of doing the other way which has led me into a situation where I was missing several details

Also, I have learned that it is also important to deal with errors immediately. Trying to deal with multiple errors at the same time gave me a difficult time to solve those. It could partially be because my plan to attack this project was not perfect, however, I have learned that dealing with them early and often actually gives me more opportunity to look over my plan and find areas to improve.

In conclusion, I have learned to start implementing after being as prepared as possible. However, during implementation, I should be aware that it is ok to make changes as I make progress and it is better to make those while they are small.

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

As I mentioned, I would start with better plan and feel more prepared before implementing. I would have allowed myself more time to prepare the plan to attack this problem rather than figuring out the plan as debugging and implementing. The main reason why it took me very long to figure out what the plan to attack was that I have tried to figure out how every component would work together at once, rather than thinking them separately. Yes, when I have coded them, I have specifically focused on the class that I was implementing. However, I did not take the same approach to coming up with plan and tried to go with fully prepared plan.

Also, if it would be possible, I would like to challenge myself with group question as well. I have some experience in managing a project by myself, hence it would be quite an experience if I was able to work on group question as well.