The design of the project is basically the same as what I said in the plan. The projects are separated into few parts. The design pattern I used is the observer. Each player is an observer of the game (game includes the information of table and deck). Each time when the game has changed, we notify the observer and update their current status (update each player's legal_play which is a vector of strings recording what legal cards can the player play). The main function defines the game, and the 4 players, and then shuffle the cards, and start the game. The loop of different round when the maximum score is less than 80 is also controlled in the main function. To let the players doing something in their turn, the program jumped to the player class. In the player class, excepts some basic functional function, the most important thing is doTurn() and notify(). The function notify() is used for update the player's legal_play, and the doTurn() is used for the details of the player's turn. In the doTurn(), the function first judge whether the turn is the first turn (if yes, then the first play have to play 7S, and other 7 are not legal, which is different from the implementation of the notify function, so I chose to separate it as a special case in the doTurn function). Then based on the player is computer or human, the function do different things. Computer is just choosing the first element of its vector. For human player, I add a loop to allow player do multiple command in one turn. Also, there is some judgement to ensure the human player obeys the rules. After each turn, we notify the observers in the main function. The calculation and conclusion of the game are also in the main function, once at least one of the total score of a player is greater or equal to 80, the loop will stop automatically, and the program will terminate.

For the question provided in the staraight.pdf, the design pattern that I chose to use is the observer pattern. The reason is stated above.

For the first questions that listed in "5.5 Players", to allow computer players, in addition to human players, the player structure has a Boolean variable in the class, which represent the Player is a human or a computer. And each time when the doTurn() function be called, we firstly check whether the Player is a human or computer, and then do the corresponding things. To allow computer players have different strategies, we can also add a variable to store the strategy. And this strategy only works when the Player is a computer, and may be changed while the Player be notified.

For the second question that listed in "5.5 Players", to allow human player easily to leave the game and transfer the Player into computer, as I mentioned in the former question, there is a Boolean variable represents the Player is a human or a computer. Therefore, when the ragequit command is received, program will set this variable to be representing the computer player. Since the doTurn() function will check what type of player is the Player is, the transfer is already finished(strategy may also be initialed, so every essential information for a computer player to continue the game is already exists).

The lesson that I learn about writing large programs is that separate the program into different piece to maximize cohesion and minimize coupling is extremely important. It will decreases the time need to spend debugging and revise the program.

If I had the chance to start over, I may remain what I did in this program. Because the intuition to solve this problem and the approach to solve it is basically correct. The only thing that may need to improve is some function that I use too less time.