ECS769P - ADVANCED OBJECT ORIENTATED PROGRAMMING - 2017/18
Program report

Name : Deni Setiawan
Student ID : 170357407

Introduction

This is a simple card game called "pontoon". The game aims to collect a hand of cards that sum to a value as close as possible to, but not exceeding 2. This game is implemented in C++. This report will show the design of the game, data structure used, and the design of the classes.

Main Requirement

1. The game consists of two players, player and bank.
2. Each player holds maximum 5 cards, with 2 cards in the beginning game.
3. Player has option to switch and twist.
4. Player should collect a hand of cards that sum to a value as close as possible to, but not exceeding 21.
5. The player wins if there are less than 5 cards in hand, total value less than or equals to 21, and the total value in hand is greater than the bank.
6. If both the player and the bank have the same value, the one has more cards in hand (maximum 5 cards) wins. If they have the same value and same number of cards, it is draw.
7. System can save the history of the game in txt log file.

Compilation Instructions

- I use code below to compile

```
g++ -W -Wall *.cpp -o 2go -std=c++11
```

- If put_time error occurs, please check the compiler version by typing g++ --version. This program is working well in my mac computer with g++ version 4.2.1.

Limitations of game

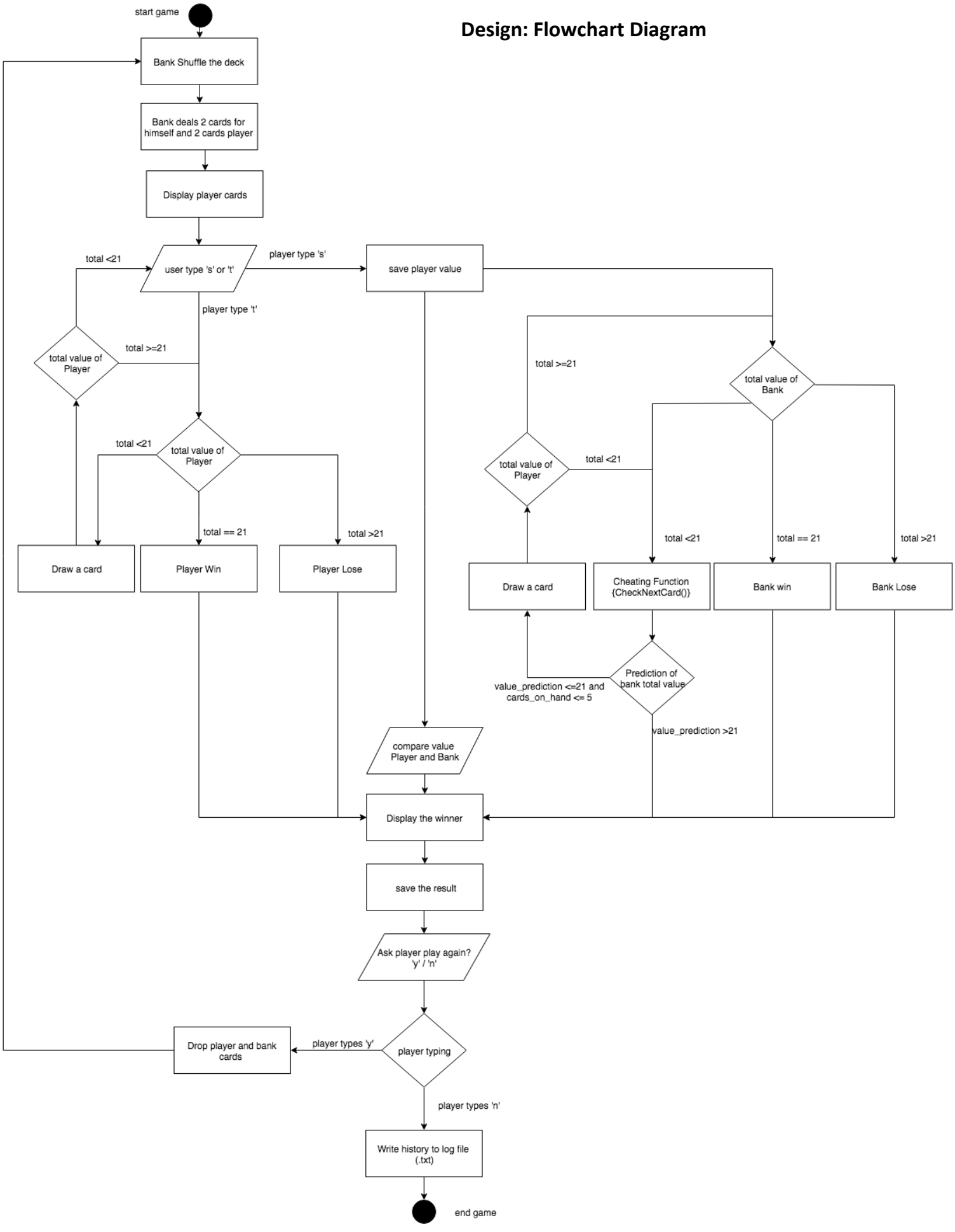
- The draw condition is not counted in log file, but it still appears in history detail.

Design of the game:

Encapsulation

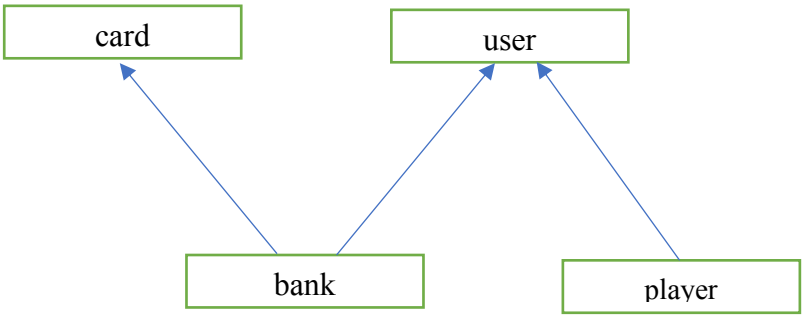
Mainly, the visibility of all methods in this project is public while the attributes(variable) are public. This makes no direct manipulation of a specific variable. Moreover, pass by reference is always used to make sure that there is no variable copy when calling a method/ function.

Design: Flowchart Diagram

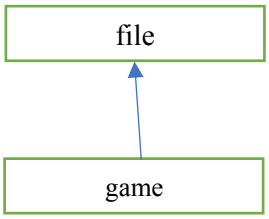


Design: Class Inheritance

Player and card:



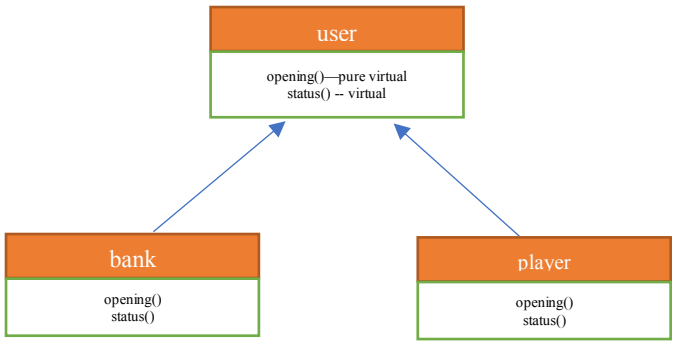
Game:



Explanations:

1. The card is base class for bank class, as the requirement that bank should be able to shuffle and the deck. Hence, the bank class has multiple inheritances from user class and card class. This may be controversial, but in my opinion, it will be safer and need an instance of an object that can be used multiple times. Hence, it may increase the performance. Another way is directly manipulating in the main program, but it will increase the complexity of the logic. Thus, I rejected it. Also, this will be very useful for the bank to run his logic and to run 'cheat' function which can see the next card. Without this function, the bank may be not able to predict the card and to challenge the user.
2. User class is base class for bank class and user class which implement robust inheritance and polymorphism. The player and bank have same number of maximal cards on hands and can count their value on the cards.
3. File class is base class for game class. This file class contains full manipulation of file such as creating a txt file, get date, and saving the history of the game, while the game class is able to manipulate whole game, so it is better to combine this feature to make the functions easy to call when in coding phase.

Polymorphism



- Opening method is for indicating user action. For bank, it is used for indicating that he draws a card. the bank needs to give information when he draws a card while the player not. That is why the player method is empty.
- Status method shows who is the turn. The interface of bank is different with player. Hence, using same function it will be useful to override the function and change the output.

Data Structure

Vector
Vector is used to store the card on the player and bank hand. The main consideration is because it will be flexible in terms of size and it needs inserting and removing objects frequently, so using this will be beneficial. To optimise this, instead of using push_back, I use emplace_back. It will avoid extra copy or unnecessary operation. Another advantage is, by using standard library system can show the size of the vector without creating additional function.

Array:
I used array for the card deck because it has fixed number of 52. Using array also can make access time constant and relatively fast compared to other data structures. Although it needs a variable to store the position of current array when player or bank draws a card, in my opinion, it will not be a big deal.

For the shuffle algorithm, I used simple algorithm. Firstly, I generate a random number which is combining with time to make sure that the number produced is unique. Then I use modulo by 52 to make sure that the number produced no exceeds 52 and put it in the array.

Set:
Set is used for validating the user input. In this project, there are two user inputs the first is when user will decide to stick or twit and the second is when the game is over, then the player is asked to play or quit the game. All correct answers are stored in string set container for example ‘y’, ‘n’, ‘s’, and ‘t’. Then when the user types something, the system uses find() to find given input. If given input match with value in the set, another function will be called to execute. Nevertheless, if the user types inappropriate character or wrong input, the system does not execute anything and only asking for input. The benefit of this system is that it does not need to consider the user input whether the data types or character length. Hence, when the user types wrong input, the game can keep safe. Although there is another way to face this issue such as using template. I considered to choose this method because we can also add other keys to the set more easily for example instead of solely using ‘y’ for input, we can also add ‘yes’ or ‘Y’ to make the game more user-friendly, and it will not affect the system significantly.

Template:
In this project, the template function is only used in count the won and lost of player. Because, instead of using the standard library, it will be more efficient to use a simple function to count the total won and lost. Moreover, it will be more flexible because we can change the form of data type easily for example, right now, the return data is integer, but it is easy to convert to string if it is necessary.

Exception Handling:
The most crucial error probably will occur when the game tries to write a log txt file. This is because the program will be run in different environment and operating system. Hence, try and catch function is implemented in this function so that the system can throw an error message when an error occurs.

Possibility to use another game:
The class card is open to use another game such as solitaire, blackjack and poker. Because it has same structure of card, but the difference is the rule of playing. This class can detect the position of the card by manipulating the current_card variable. Then, many functions such as shuffle function, converting function will be useful to reuse.