Project Deliverable 2

SYST17796

Fundamentals of Software Design and Development

Submitted to Professor Ahmandeep Sidhu

Submitted by

Group 4

Jean/Tran Hoang

Robin Methot

Alexander Matthew

July 31, 2020

Table of Contents

I. Overview

The chosen card game to deliver for the course's group project is BlackJack (also known as Twenty-One). In order to win the game, the player has to earn a point as close to 21 but not exceeding it.

II. Project background & description

The game is generated with a standard deck of 52 cards that has 4 different suits for each 13 ranks, no jokers included. Each cards have a corresponding point of their rank/value. Face cards value 10 points and Aces are worth either 1 or 10 points.

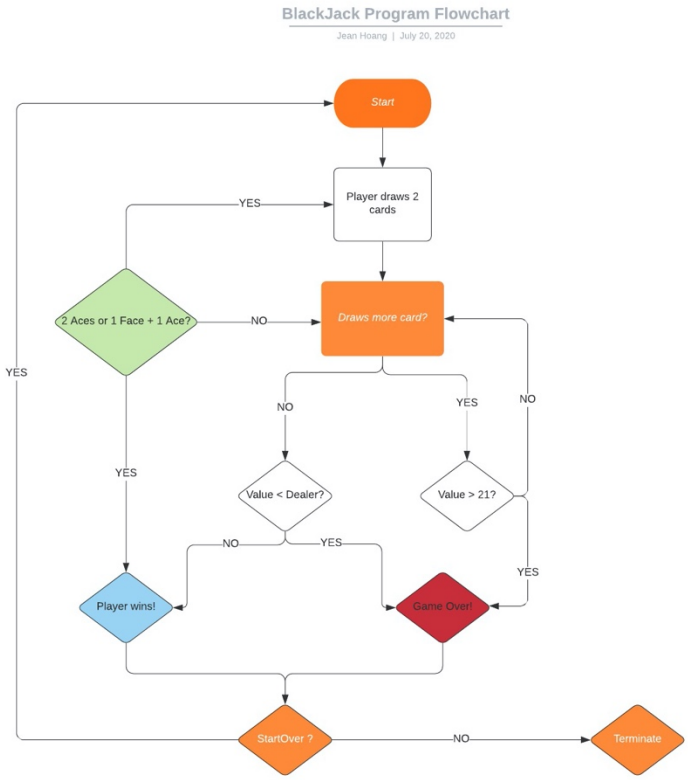

Figure 1 BlackJack flowchart from Player perspective

The program will start up with a welcome message then it will output the first round of drawing, where dealer/player will draw 2 cards. The dealer's deck only shows 1 card while its second card remain hidden. The player's deck is shown with the total value. Player gets to go first.

Player is then prompt if he/she wants to draw more cards, if player said yes:

If the cards exceed 21 on the next draw, player loses immediately, the program will prompt a "GameOver" message then it will ask if the player wants to start over. If player chose no, the program is terminated.

If the cards value is less than or equal to 21 and player chose to draw again and the player's deck value is greater than the dealer's. Player wins, the program will prompt a

"You Win" message then it will ask if the player wants to start over. If player chose no, the program is terminated.
If player chose not to draw after the first initialization of the game and the card's value either greater than the dealer's value or equals to 21. Player wins, the program will prompt a "You Win" message then it will ask if the player wants to start over. If player chose no, the program is terminated.
At the first round of drawing card, if the player draws 2 cards that are both Aces or a combination of an Ace and a face card, player automatically wins.
To reduce the redundancy for the report, the same logic applies for the dealer, the only difference is dealer's choice will be randomized.

A. Use cases diagram

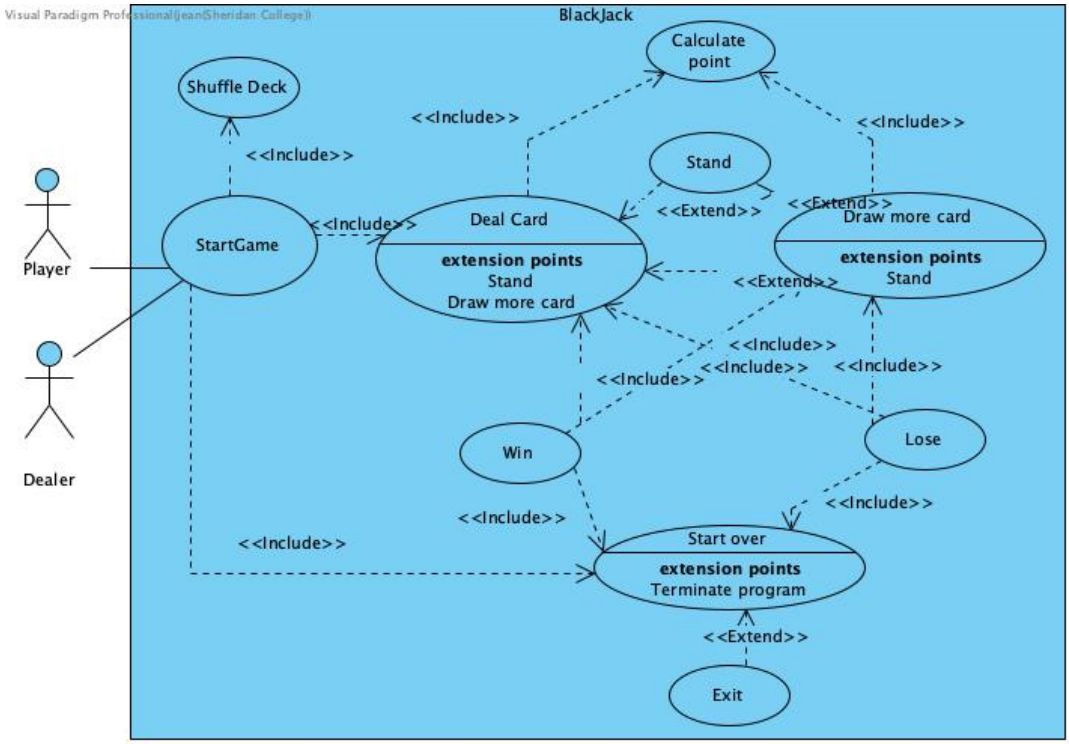

Figure 2 BlackJack Use Case diagram

B. Narrative

1. User starts the program
2. Program displays Welcome message and deals 2 cards for player and dealer
3. Program checks if user wins from the first draw
   3.1 User wins from first draw
      3.1.1 Program prompts "You win" and asks if user wants to start over
      3.1.2 Program prompts "Game Over" and asks if user wants to start over

3.2 User doesn't have special case and the cards values are less than 21

4. Program asked if user wants to draw more cards while having 1 of dealer's card facing up
   4.1 User chose "Hit"
      4.1.1 User total point exceeds 21: program prompts "Game Over" and asks if user wants to start over
      4.1.2 User's total point is still below 21: program prompt if user wants to draw more cards again
   4.2 User chose "Stand"
      4.2.1 User's total point is greater than the dealer's point, program displays "You win" and prompt for StartOver
      4.2.2 User's total point is less than the dealer's point, program displays" GameOver" and prompt for StartOver
5. Program asks if user wants to StartOver
   5.1 User wants to play again
      5.1.1 Program restarts the game
   5.2 User wants to quit
      5.1.2 Program prompt goodbye message and exits the game

C. Class diagram

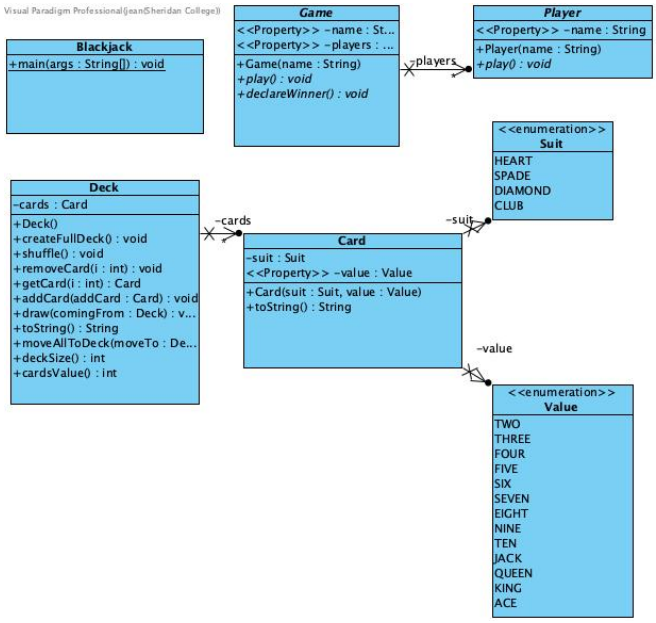

Figure 3 BlackJack base class diagram

BlackJack is the Main class for the entire program.

The Game class models the BlackJack game, in which it includes a String contain the name of the game (BlackJack), a Play method where dealing cards and game logics are applied and a declaration for winning the game method, the startover prompt will be included in this method to ask if user wants to play the game again. The Player class will store game points and cards for both the user and dealer. It includes methods to calculate points and shows what the cards the users are having. This class is a child class from the Game class where it inherits the declaration for winning the game method if user's or dealer's lose to the conditions of the game.

The Deck class is used to generate cards for the Game class. Deck class obtains Cards from the Card class where it gets the Values and Suits from a standard deck of card from the 2 enum classes Values and Suits. The Value enum contains constants for 13 cards that corresponds to its value with the face cards holding values of 10 points while the Suit enum contains constants of 4 Suits of Heart, Diamonds, Clubs and Spades. After generating a deck from the Card class, Deck class then shuffle to ensure players get random values for their dealt cards. The Deck cards also include a get/set method for drawing cards.

D. Domain class diagram

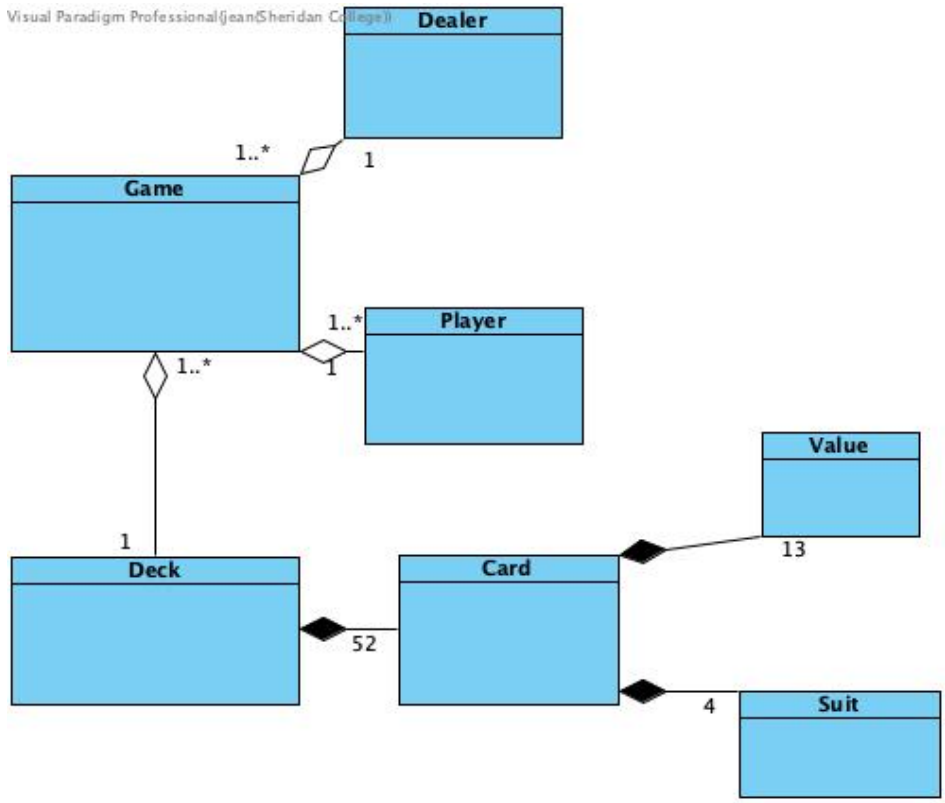

Figure 4 BlackJack Domain Class Diagram

III. Object-oriented Design Principle

Encapsulation

To ensure encapsulation for the program, all data members used to generate the game are private, they cannot be accessed from anywhere else outside of the class. Accessors and mutators methods are used to retrieve those data. Only the members that are used to prompt into the program's interface are set to public. Ensuring encapsulation for BlackJack is highly crucial since players are not supposed to know which cards they will withdraw if they chose to draw more cards on their turns. By making sure fields are private, users will have no way to tell the logic behind the program to cheat on the game.

Delegation/Cohesion/Coupling

The program aims for high cohesion and loose coupling in between classes. Each class has little dependency to one another so that when changes are made, they only have to be modified from one instead of multiple locations. The program could have been made with only 4 classes, however, the Player class, Rank enum and Value enum are added so the program is easier to comprehend.

Flexibility/Maintainability

By making the program highly cohesive and loosely coupled, the program is more flexible. In the future, more feature can easily be added into the programs. In example, the program could let the user bet money for the BlackJack game and if users ran out of money the game will terminate. This feature can be included in the Game class. Another example is if users with to have more players, the program can include more players with specific ID through Player class. The new features are constructed at ease since they only need to be constructed in their appropriate class with very little changes to the other classes. In the above examples, only Game and Player classes need to be modified.

Aggregation

The Game class has an aggregation relationship with the player, game and deck class. For this instant, there will only be 1 deck of card, 1 dealer and 1 player for each BlackJack game.

Composition

The Deck classed is composed by the Card class and the Card Class is composed from the enums Value and Suit. There are 13 values for a standard deck of card for each 4 different suits. This makes up 52 Cards that belong to 1 Deck of card. A Deck of card would not exist without these values.

*Side note

The UML Class Diagram might be subjected to changes later since our group are considering adding bet money into the program.