Project 1: Blackjack Game

CIS-17C-47065
Name: Gregory Truong
Date: 4/23/2023

Introduction

Title: BlackJack

- Rules of the game:
    - The dealer will ask how much you are willing to bet, you will start a number of coins you can deal.
    - Once you have placed your bet, the dealer will begin outputting cards that you will have.
    - You can choose to either hit, to gain an extra card to get nearer to 21, a chance double down and double your betting money, allowing you to get more money if you win, but you cannot hit again, or stand to not receive anymore cards.
    - The dealer will show his/her cards to you
    - If the dealer wins, you will lose the coins you bet. If you win, you gain double of the amount you betted. If you win off of the double down, you will win quadruple the amount you betted.
    - The game will stop when either you are out of coins or when you decide to stop the game.

Summary:

This project took about almost all of my time during the past week of doing this project, around 150+ hours of work. I kept getting hiccups after hiccups and eventually got it up to working speed at the end of the deadline. I programmed most of the work around the basic functions although I still haven't gotten around to making it function as well as I thought it would. It eventually broke down till it wasn't fixable and eventually looked nothing as it was before. I still think this is fixable though as I just need more practice with the project. Just some more tinkering and working out kinks around what I should be doing. You can find the project in the folder labeled "Project 1" and the folder "Project 1 Blackjack" along with the write up.

Description:

The main point of this project is too practice the skills of what I learned and put it into practice into making my very own game.

Pseudo Code:

Start

Player bets an amount
    Dealers starts the game

Dealer starts his/her turn
    Game Started
        Players is given 2 cards
        Dealer gets a card

Player decides to hit
    Dealer will reveal his cards

Else Player decides to Double Down if the chance is offered
    Dealer will give one card
        Dealer will show his card

Else Player Stands

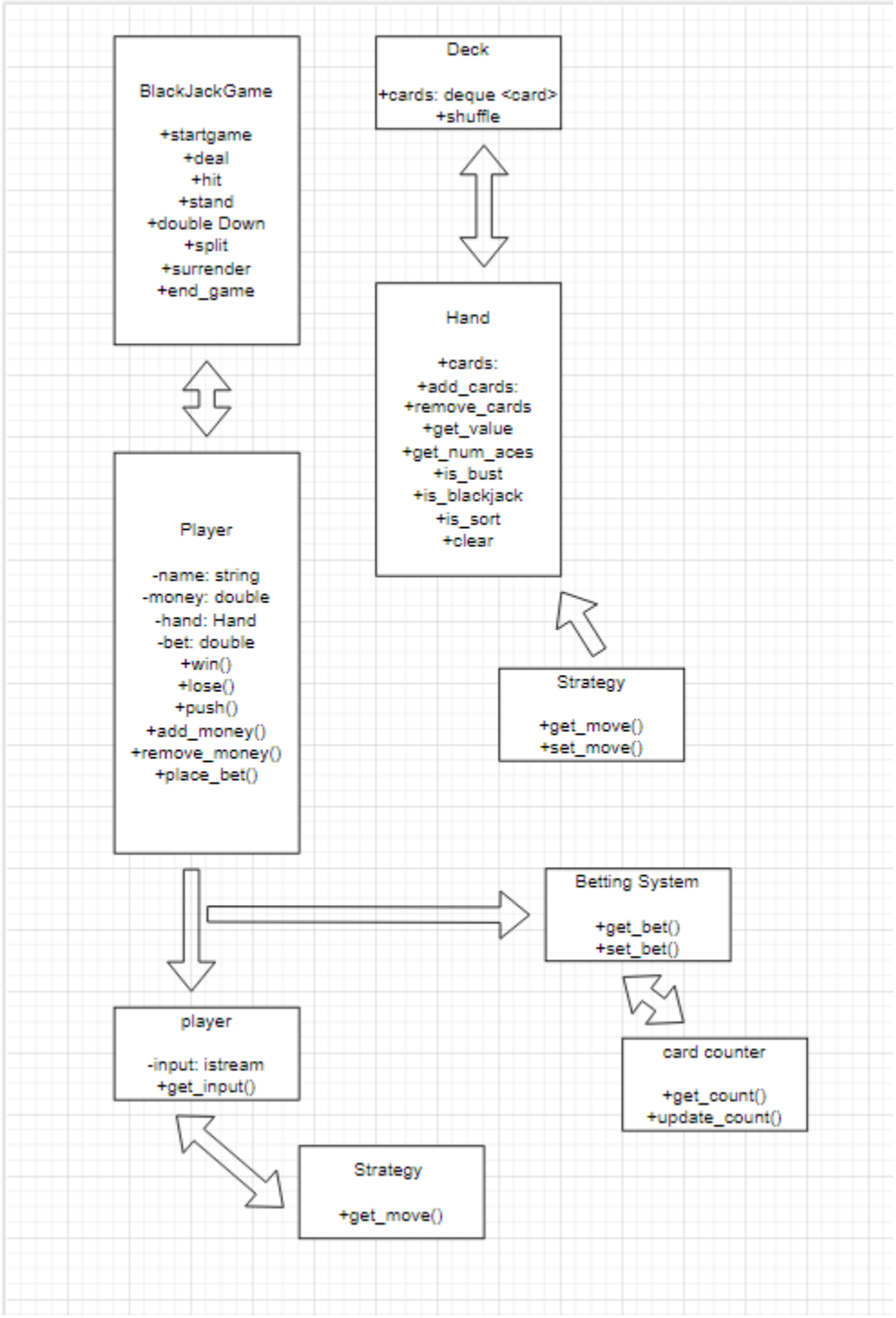
Dealer will not give player a card
Dealer will show his card

If Player Wins
Player receives double the amount of coins

If Player Loses
Player loses
If Player doesn't have anymore coins, the game ends

Loops until the player decides to play again or not

UML Diagram


BlackJackGame
+startgame
+deal
+hit
+stand
+double Down
+split
+surrender
+end_game
Deck
+cards: deque <card>
+shuffle
Hand
+cards:
+add_cards:
+remove_cards
+get_value
+get_num_aces
+is_bust
+is_blackjack
+is_sort
+clear
Player
-name: string
-money: double
-hand: Hand
-bet: double
+win()
+lose()
+push()
+add_money()
+remove_money()
+place_bet()
Strategy
+get_move()
+set_move()
Betting System
+get_bet()
+set_bet()
player
-input: istream
+get_input()
card counter
+get_count()
+update_count()
Strategy
+get_move()

Checkoff Sheet Contents

1. Container classes
    a. Sequences
        i. List
            1. For a list, I used it to display the deck of cards and showing what the card is, what number it is and how the player wants to decide how to use there cards that they get
    b. Associative Containers
        i. Set
            1. Set keeps track of the cards being dealt and how they are being played. This prevents from any card being dealt more than they should.
        ii. Hash
            1. Hash stores the information of a player like how many coins they have and how many they have won or how many games they have played.
    c. Containers adaptors
        i. Stack
            1. Stack will represent the cards that have been dealt and already played with so that the card doesn't show up again when a new game is being played.
        ii. Queue
            1. Queue will represent the turns each person has between the dealer and the player so that each person's turn does not over 1.

2. Iterators
    a. Concepts
        i. Trivial Iterator
            1. This iterator could access each card one by one
        ii. Input Iterator
            1. This iterator allows the player to input what they want like accepting cards, playing again, or betting the amount they want
        iii. Output Iterator

1. Output iterators can write down what cards are being dealt to the player and what cards the dealer has himself in the console

iv. Forward Iterator
1. Forward iterators make it so that iterators can iterate the cards in the deck or over the players' hands.

v. Bidirectional Iterator
1. You can use this iterator to shuffle the cards in a deck by swapping two positions of two cards, giving it a shuffled appearances in the code

vi. Random Access Iterator
1. You can use random access iterators to deal the card to a specific player.

3. Algorithms
   a. Non-mutating algorithms
      i. Count
         1. Count allows the counting of the cards and how many cards are either in the deck, in the players hands, or in the dealers hand
   b. Mutating Algorithm
      i. Random_shuffle
         1. This allows the cards too be shuffled in the deck and allow the console to produce a different output each time either the player hits or double downs for a card.
   c. Organization
      i. Sort
         1. This allows the console to determine where goes where and helps the console determine the discard pile and where that card should go when the card is being dealt with.