**Project 1**

**<Blackjack>**

**CIS-17C**

**Name: Jesus Ferrer**

**Date: 04/26/2023**

**Introduction**

The main game for the project is Blackjack, but there are 3 other options to choose from. I chose Blackjack because I wanted to learn how to play Blackjack and thought this would be a good time to learn. The 3 other options that user can choose are 3 different ways to display deck of cards. I spent around 15-20 hours on the project, and there are about 850 lines. I created 5 classes, which are “BlackJack”, “Player”, “Dealer”, “Deck”, and “Card”. The project can be found in the “Project1-BlackJack” folder using the following link <https://github.com/suusej/23Spring-CIS17C>.

**Approach to Development**

The concepts I used in this project were list, set, map, stack, queue, random access iterator, output iterator, forward iterator, find, random\_shuffle, and sort. The main problem I ran into is fitting concepts into the program without making it inefficient. This the reason why I added other choices for the user in this project. The program is definitely not efficient, but it works properly and meets the requirements of the project.

**Game Rules**

The rules for this program of Blackjack are pretty similar to the official game. This program only runs one round, but you can have up to four players. After entering the players’ name and bets the cards will be distributed and displayed. After the cards are displayed, the dealer’s will draw cards until their hand is 17 or higher. Next each player will take turns to play against the dealer. If both the player and dealer bust, the dealer wins, if the player is closer to 21, the player wins and same thing for the dealer. After each player has played, all players that won will be displayed.

**Description of Code**

I created 5 classes in this program. The classes are “BlackJack” which holds all of the game’s main objects like players, deck, and dealer, then there is a “Player” and “Dealer” class. I made the “Dealer” class inherits from the “Player” class because both the dealer and player are players in Blackjack, but the dealer also has access to the deck. The last two classes are “Deck” and “Card”. All of the classes have a .h and .cpp files, except for “Dealer”. “Dealer” is included in the “Player” files.

**Sample Input/Output**

**Menu with option 1**

Table

Description automatically generated

Table

Description automatically generated

**Round of Blackjack**Table

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

**Option 3**Table

Description automatically generated with medium confidenceTable

Description automatically generatedText

Description automatically generated**Option 4**Table

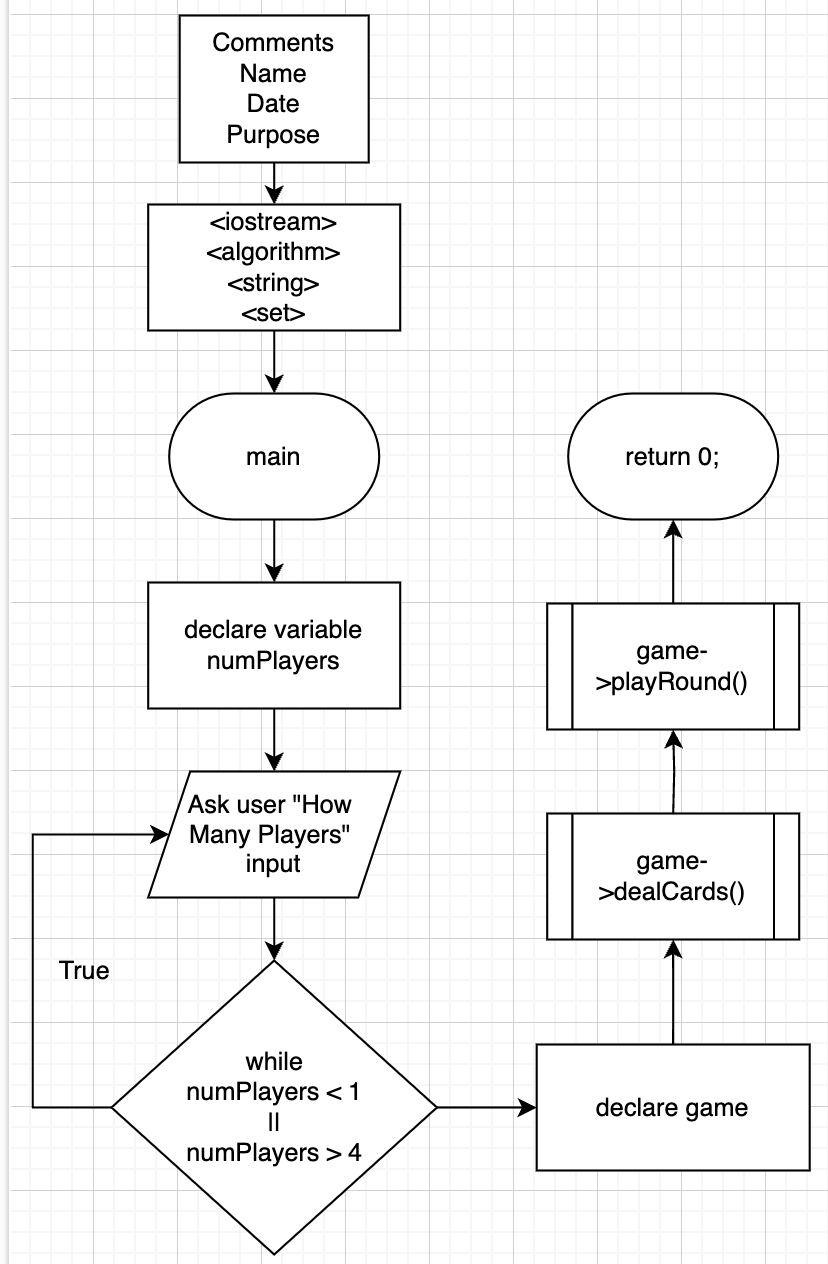
Description automatically generated

**Check Off List**

**Table

Description automatically generated**

**Flowchart**

****

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Pseudocode**

numPlayers = 0

print "Enter the number of players (1-4)"

read numPlayers from standard input

// Validate the user input

while numPlayers < 1 or numPlayers > 4:

print "Enter the number of players (1-4)"

read numPlayers from standard input

// Initialize the game with the number of players

game = new BlackJack(numPlayers)

// Deal first two cards to all players

game.dealInitialCards(numPlayers)

// Deal cards to each player in the queue

for i = 1 to numPlayers:

// Deal two cards to the player

dealer.dealCard(plyrQueue.front(), deck)

dealer.dealCard(plyrQueue.front(), deck)

// Print the player's hand

print plyrQueue.front().getName() + "'s Hand"

plyrQueue.front().getHand()

print "\n"

// Move the player to a temporary queue

temp.push(plyrQueue.front())

plyrQueue.pop()

// Restore the player queue

plyrQueue = temp

// Deal two cards to the dealer

dealer.dealCard(dealer, deck)

dealer.dealCard(dealer, deck)

// Print the dealer's hand

print dealer.getName() + "'s Hand"

dealer.getHand()

// Start playing the game round

game.playRound(numPlayers)

// Dealer plays until hand value is at least 17

while dealer.getHandVal() < 17:

dealer.dealCard(dealer, deck)

// Print dealer's new hand

print dealer.getName() + "'s New Hand"

dealer.getHand()

print "\n"

// Play the game for each player in the queue

temp = empty queue

for i = 1 to numPlayers:

// Ask the player to choose an action

do:

// If not the first player, show the dealer's hand

if i != 1:

print "\nDealer's Hand"

dealer.getHand()

// Show the player's hand and ask them to choose an action

print "\n" + plyrQueue.front().getName() + "'s Turn"

plyrQueue.front().getHand()

print "Choose Action (1,2)\n1. Stand\n2. Hit"

read action from standard input

// Deal a card to the player if they choose to hit

if action == '2':

dealer.dealCard(plyrQueue.front(), deck)

plyrQueue.front().getHand()

// Check if the player busts

bust = checkBust(plyrQueue.front())

// Decide winner if the player busts or stands

if bust:

print plyrQueue.front().getName() + " Busts\nDealer Wins\n\n"

else:

if checkBust(dealer):

print "Dealer Busts\n" + plyrQueue.front().getName() + " Wins!\n\n"

else:

checkWinner(plyrQueue.front(), dealer)

bust = true

// Move the player to a temporary queue

temp.push(plyrQueue.front())

plyrQueue.pop()

// Print the list of winners

print "List of Winners"

for each winner in winners:

print winner

**Class UML**

**Diagram

Description automatically generated**