Project 2

Title

Blackjack Simulator Version 2

Course

CIS-17A

Section

43320

Due Date

June 11, 2021

Author

Annabelle Tamano

Table of Contents

1 Introduction

As the guideline for this project expect a recreation of existing card game, dice game, and/or board game to be recreated as a C++ program, I chose to build upon my first project by coding Blackjack again.

Blackjack is a gambling game that typically allows multiple players, adopting rules that vary from where they are played. For the most part, I chose to base the project off the basic rules of the game. However, because this project is intended to be a text-based simulation, there are some limitations that I am working around through the process of developing the program. Like the first version of my project, I kept these significant changes to the Blackjack game, as follows: there are no doubles or splits, and the cards are represented by certain characters rather than pictures of full cards, omitting information such as the color of the card.

While I indicated that the game would remain the same as the first project, I have made certain changes to this new project to make it stand out on its own. First, I obviously am utilizing more of the discipline of object-oriented programming to build this project up. Next and last, I am allowing for the user to play against up to three opponents to add to the complexity of the game.

2 Game Play and Rules

The user is sent to the main menu where they are given three options: to see the rules for Blackjack (‘1’), to play the single player game (‘2’), to play the multi-player game (‘3’), or to see the scoreboard (‘3’). The user then must enter ‘1’, ‘2’, or ‘3’ to indicate which action they would like to happen. After each action is executed, it sends user back to the main menu. To exit the program, the user must press any other key than ‘1,’ ‘2,’ ‘3,’ or ‘4.’

When entering the game, the program randomizes the number of players in the game, giving the user anywhere from 1 to 3 opponents. If the user enters a game with opponents, their opponents’ names and starting bets are generated. The program will then prompt the user for their name and their starting bet and begin the round of Blackjack.

2.1 Blackjack Rules

- The overall premise is to beat the dealer's hand without going over 21.
- To start the game, you must place a bet at minimum $2 and at maximum $50.
- Certain cards are worth different values. Face cards (K, Q, J) are worth 10. Aces are worth 1 OR 11 (Whichever adds to a better hand). The rest of cards are worth the number that they display.
- Both you and the dealer start with two cards, however one of the dealer's cards is hidden until the end.
- At each play you have one of two options. First, you can 'Hit' which is just asking for another card. OR you can 'Stand' which signifies holding your total and ending your turn.
- If the value of your cards goes over 21 you bust. Thus, you lose all the money that you bet.

- If you are dealt 21 from the start, you got a blackjack! Therefore, you automatically win the pot of all the bets.
- The dealer will hit until his/her cards total 17 or higher. Both your hands will be compared, and if you beat the dealer's hand you win back 1.5x the value of your original bet. Else, you lose the value of your original bet.
- If you tie with the dealer or if your dealer busts, you neither win nor lose, and you get no money back.
- NOTE: To simplify the game, this version does not deal with splits or doubles.

2.2 Card Display Format

- Each card will be printed encapsulated by brackets [].
- First, the card name {A (Ace), 2, 3, 4, 5, 6, 7, 8, 9, T (10), J (Jack), Q (Queen), K (King)} will be printed.
- Then a character representative of the card's suit {C (Clubs), H (Hearts), D (Diamonds), S (Spades)} will be printed in the bracket.
- Example: [A C] represents the card "Ace of Clubs."

3 Development Summary

This project is a text-based C++ program, utilizing the discipline of object-oriented programming with the help of certain files. I developed the project with the NetBeans 8.2 IDE, as required by this course. This project displays a mastery of concepts from Chapter 13 to Chapter 16 of the course textbook, Gaddis's *Starting Out with C++ from Control Structures to Objects*.
Total Lines of Code: 1005

3.1 Version 1

This version of the program is a dud version that does not allow the user to play the game. The development of this version isolates two main pieces to the overall project. First, I created two classes that will be necessary for the coding game later, the Player class and the Card class. Then, I developed the main menu system that will operate three main functions: seeing the rules, playing the game, and seeing the scoreboard.

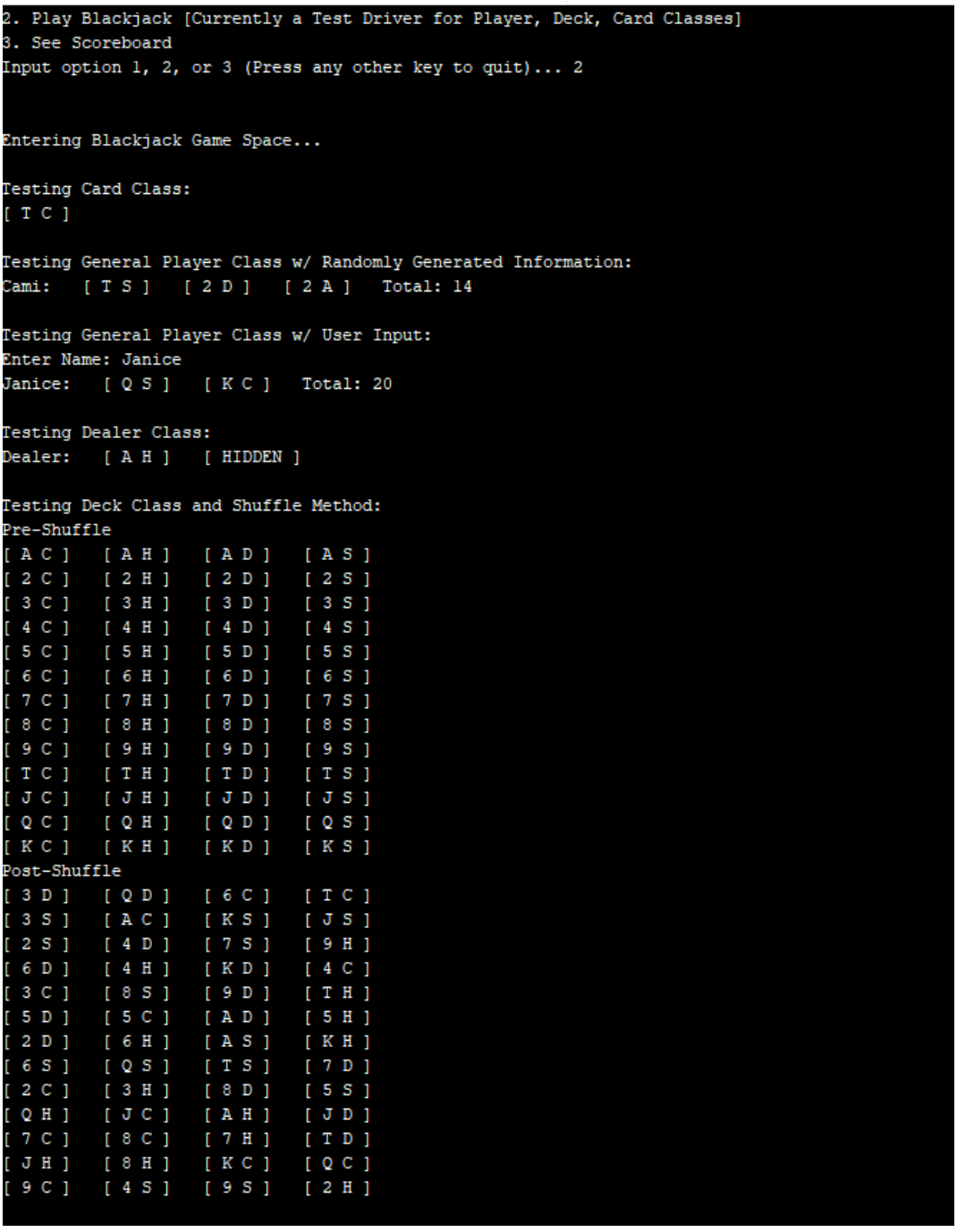
3.2 Version 2

Through the development of version 2, I began including inheritance by creating two classes GenPlyr (General Player) and Dealer, which are still both part of the Player class. However, this separation rather than a combined grouping as a player allows me to distinguish how the two function differently throughout the game of Blackjack. It still does not play the game, but it properly sections parts of the code for more efficiency.

3.3 Version 3

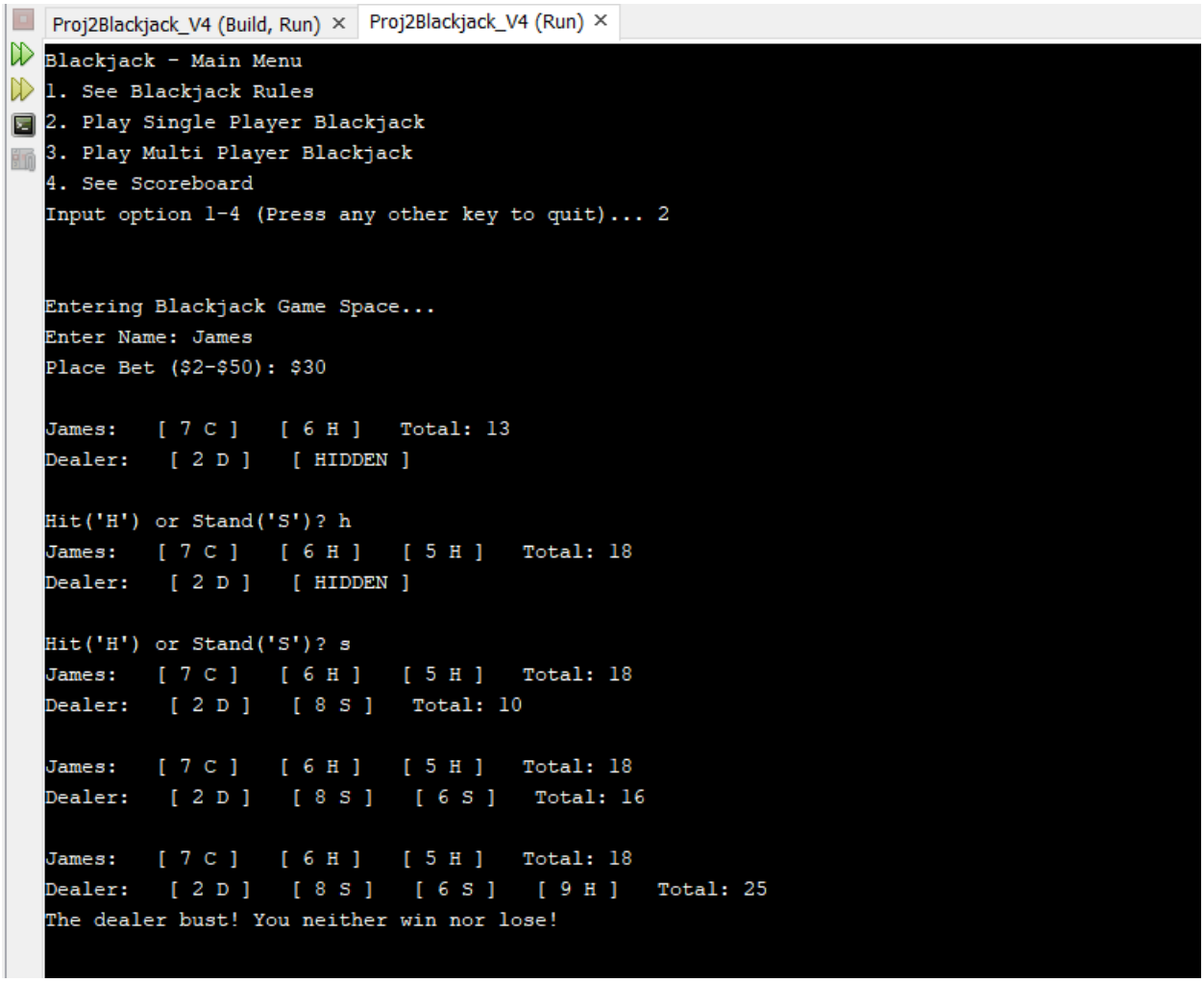
Version 3 emphasizes the creation of the Deck class to store the Cards that are crucial to the game of Blackjack. The Deck class holds two important arrays, a deck of Card and an array of indexes for the Card deck. The playG() function within the main file, tests all these new developments.

The following screenshot indicates the output of the driver tests within the playG() Function. As shown, the classes and their member functions currently work as intended.



3.4 Version 4

Moving along with the development of this project, the most drastic change in this specific version is that this version allows the user to play a single player version of Blackjack with the dealer.



3.5 Version 5

This version of the program is the start of the option that allows the user to play blackjack either as a single player or as a multiplayer. This is done through overloading the playG() function with an integer of the number of players (excluding the dealer) in the game. As well as that, to improve collecting the user's bet, this version includes exceptions to make sure the bet ranges from $2-$50.

3.6 Version 6

This version finishes of the multi-player game expanding on the original playG() function to give room for the other players in the game.

3.7 Final Version

Last, the final version cleans up lines of code and optimizes it for execution and better comprehension of the concepts implemented. As well as that, I utilized a template for the random number generator, so it would be simplr

4 Specifications

4.1 Pseudocode

The following pseudo-code was developed before the completion of the entire project. Despite this not exactly matching with the naming conventions of all parts of the program, it still represents what specific parts of the program are meant to do.

4.1.1 *main* Function

Ask user for name and store in tempN
Initialize User plyr
do {
Print the main menu with three options 1. See rules, 2. Play game, 3. See scoreboard
Ask user for option 1,2, or 3 and read in opt
opt is 1? Call prntR function
opt is 2? Call playG function
opt is 3? Call seeSB function
opt is none of the above? great user goodbye.
} while (opt is 1-3);
delete tempN
return 0 to end Function

4.1.2 *prntR* Function

Declare variables outpF and n.
Open in outpF the file “rules.txt” 7

Allocate memory into outpS
while you can read line from outpF into outpS[n] {
increment n }
close outpF
delete outpS

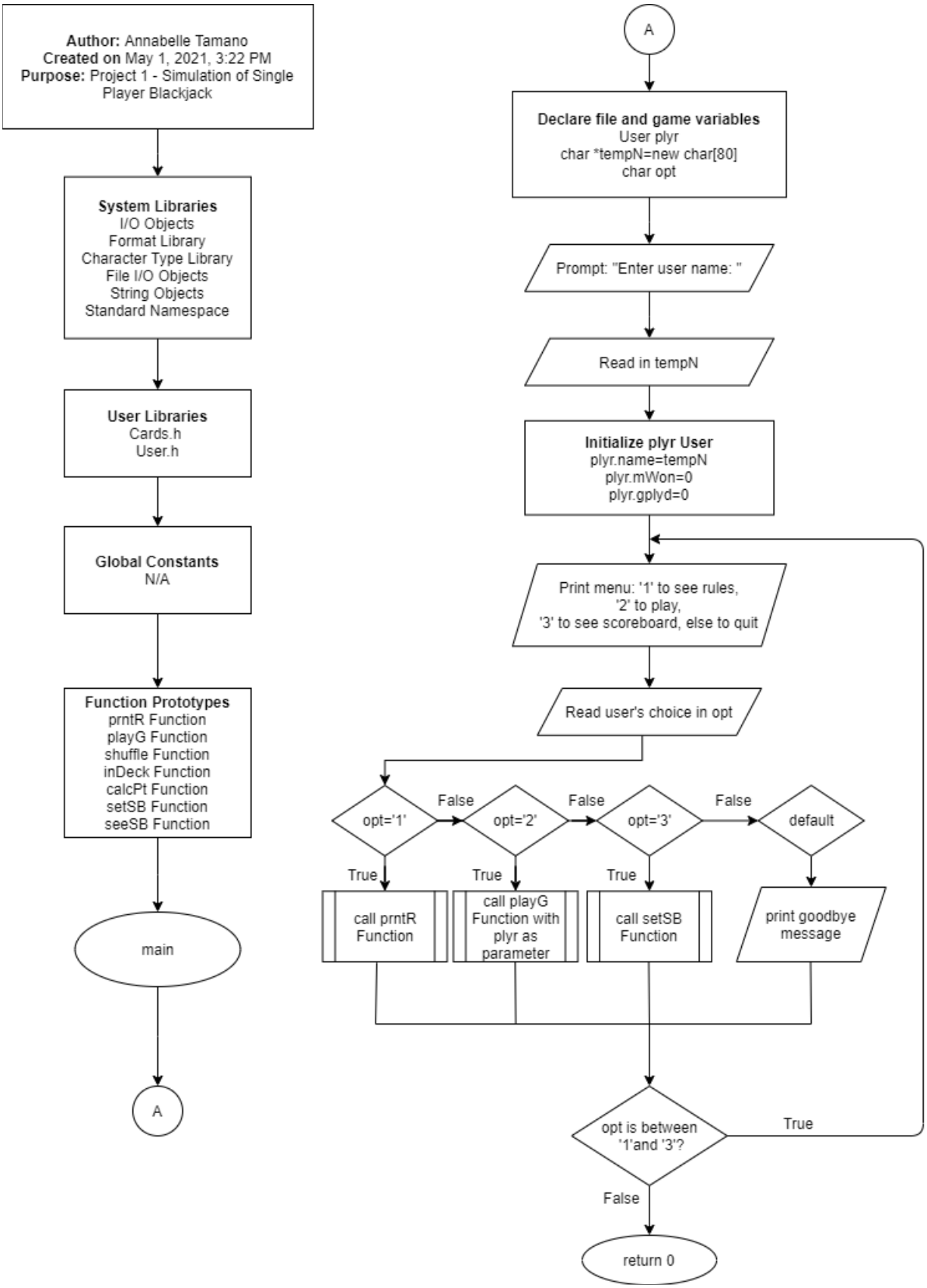
4.1.3 *playG* Function

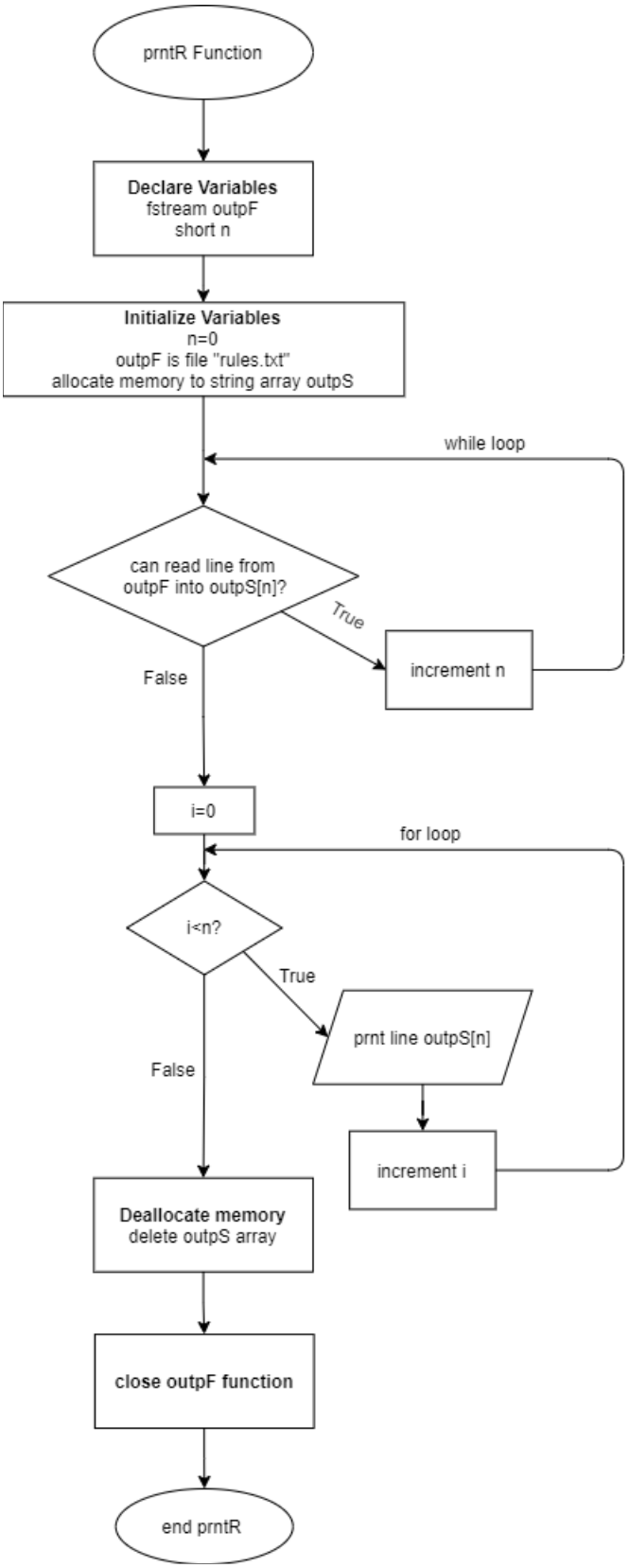
Initialize Card deck and shuffle deck indexes
Set space for user and dealer's hands
Deal the first two cards for both user and dealer
Calculate total points for user and dealer
Start game prompting user to place bet and reading in bet
do {
Use for loop to print user's deck and then print total
Print dealer's first card and indicate that second card is hidden
usrPt is over 21? User automatically loses all the money they originally bet, run ends
usrPt is exactly 21? User automatically wins 1.5x the money they originally bet, run ends
else?
User is asked to hit or stand; read in usrM
If hit? A new card is added to user's hand and total points are calculated
If stand?
Dealer adds card to hand until point value is greater of equal to 17
Compares both hands
If user's hand greater than dealer's? You win 1.5x the money you bet
If dealer's hand greater than user's? You lose the money you bet
If you tie or if they dealer busts? You neither win nor lose any money
run ends
} while run does not end
Increment gPlyd in plyr
Set scoreboard
Deallocate all the memory created for this function

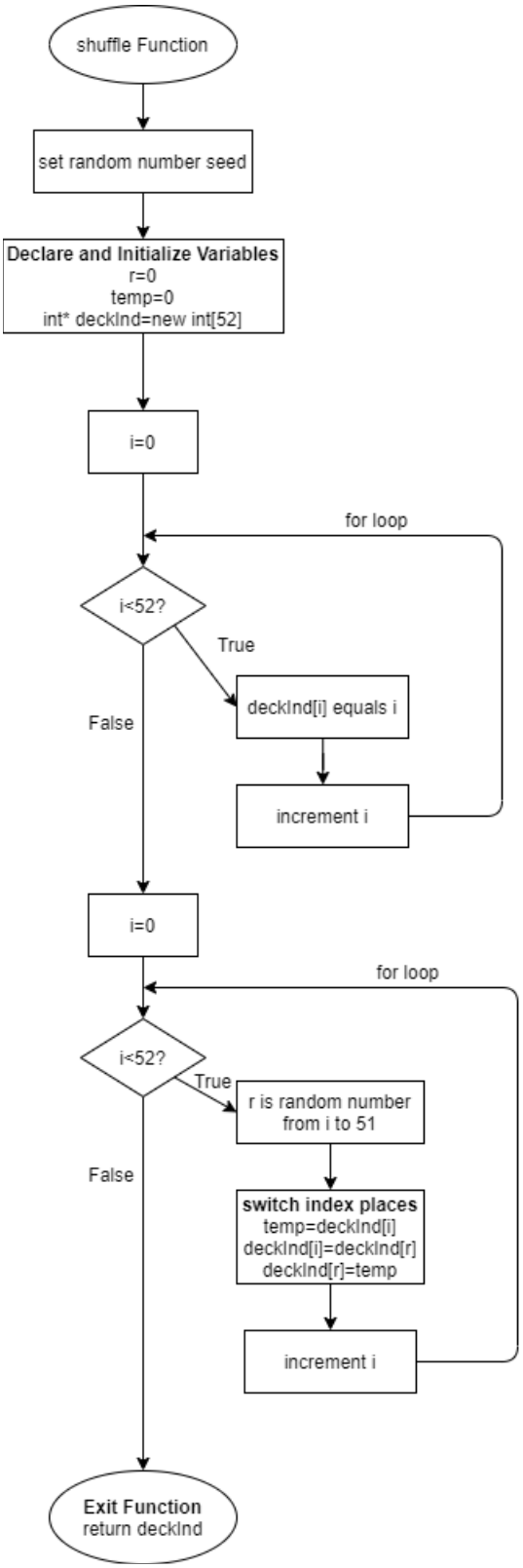
4.1.4 numGen Template Function

Set random number seed
Takes max and min from argument
Return random number between max and min

4.2 UML/Flowcharts


Author: Annabelle Tamano
Created on May 1, 2021, 3:22 PM
Purpose: Project 1 - Simulation of Single Player Blackjack
System Libraries
I/O Objects
Format Library
Character Type Library
File I/O Objects
String Objects
Standard Namespace
User Libraries
Cards.h
User.h
Global Constants
N/A
Function Prototypes
prntR Function
playG Function
shuffle Function
inDeck Function
calcPt Function
setSB Function
seeSB Function
main
A
A
Declare file and game variables
User plyr
char *tempN=new char[80]
char opt
Prompt: "Enter user name: "
Read in tempN
Initialize plyr User
plyr.name=tempN
plyr.mWon=0
plyr.gplyd=0
Print menu: '1' to see rules,
'2' to play,
'3' to see scoreboard, else to quit
Read user's choice in opt
opt='1'
False
opt='2'
False
opt='3'
False
default
True
True
True
call prntR Function
call playG Function with plyr as parameter
call setSB Function
print goodbye message
opt is between '1'and '3'?
True
False
return 0


prntR Function
Declare Variables
fstream outpF
short n
Initialize Variables
n=0
outpF is file "rules.txt"
allocate memory to string array outpS
while loop
can read line from
outpF into outpS[n]?
True
increment n
False
i=0
for loop
i<n?
True
prnt line outpS[n]
False
increment i
Deallocate memory
delete outpS array
close outpF function
end prntR


shuffle Function
set random number seed
Declare and Initialize Variables
r=0
temp=0
int* deckInd=new int[52]
i=0
for loop
i<52?
True
deckInd[i] equals i
False
increment i
i=0
for loop
i<52?
True
r is random number
from i to 51
False
switch index places
temp=deckInd[i]
deckInd[i]=deckInd[r]
deckInd[r]=temp
increment i
Exit Function
return deckInd

4.3Concepts Used

All highlighted concepts are ones that are focused on within the program. I was able to incorporate most, but some did not make sense to include into the project.

Chapter 13

- Instances of a Class
- Private Data Members
- Specification vs. Implementation
- Inline
- Constructors
- Destructors
- Arrays of Objects
- UML

Chapter 14

- Static
- Friends
- Copy Constructors
- Operator Overloading
- Aggregation

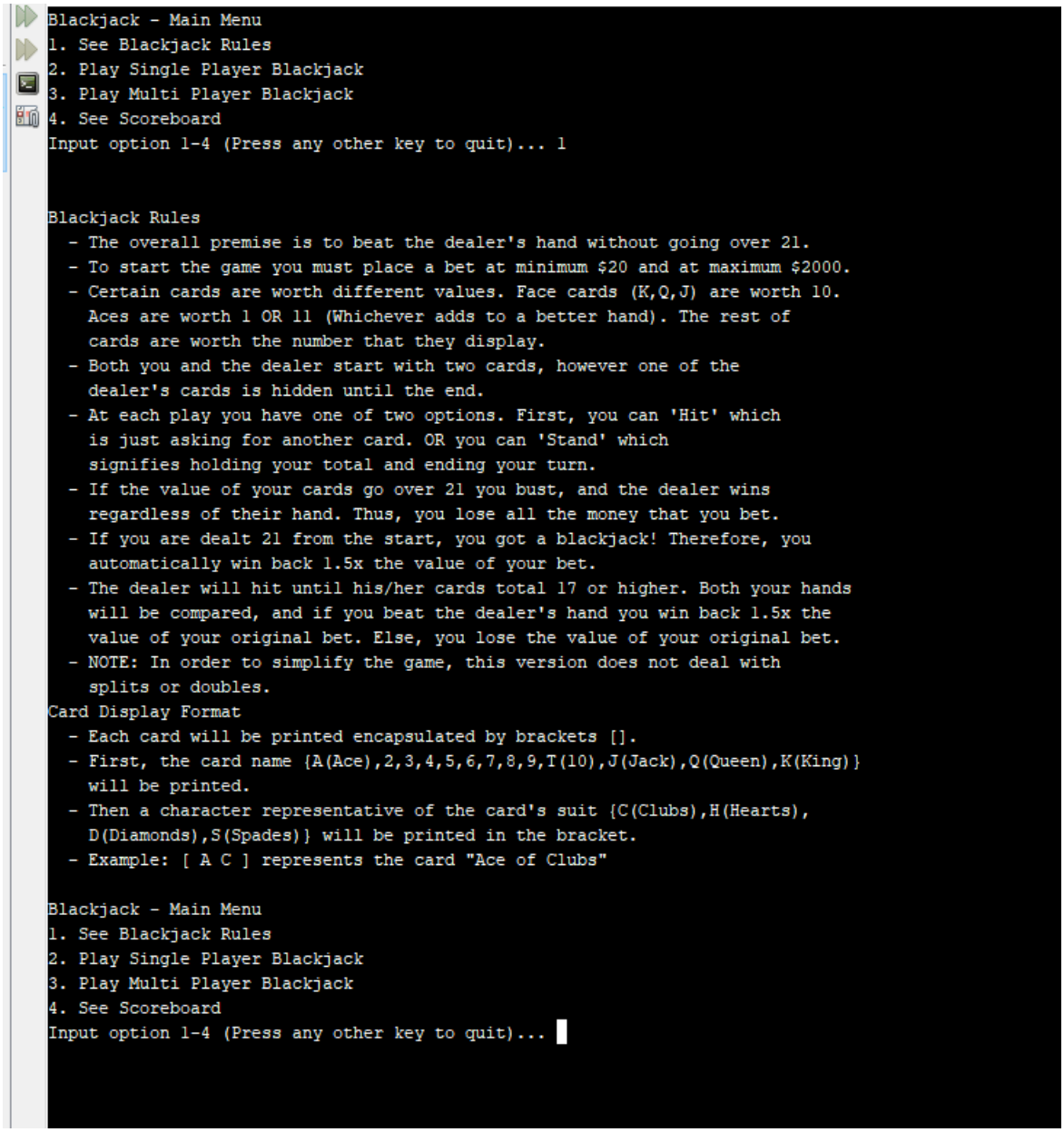
Chapter 15

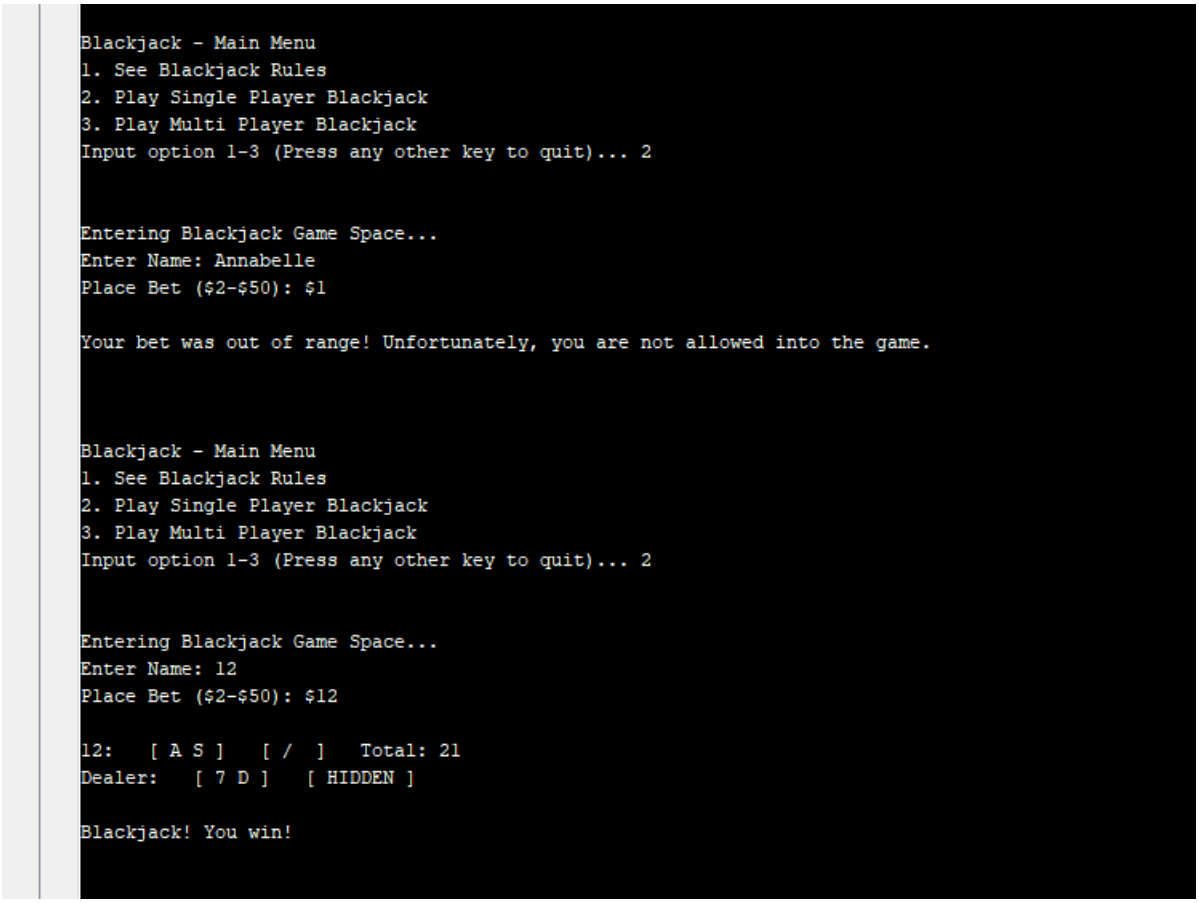
- Protected members
- Base Class to Derived
- Polymorphic associations
- Abstract Classes

Chapter 16

- Exceptions
- Templates
- STL

4.4Sample Output





5 References

To refresh my knowledge of the basic rules of Blackjack, I utilized the following website: http://www.hitorstand.net/strategy.php

For more information on how Blackjack is played with multiple people, I used the following resource: https://bicyclecards.com/how-to-play/blackjack/

I utilized the following reference to understand a basic algorithm on how to shuffle the deck of cards: https://www.geeksforgeeks.org/shuffle-a-deck-of-cards-3/