Final Project

Blackjack

Christian Villanueva
CSC 17A - 49213
18 December 2020

Introduction:

I have created a program that runs a simple version of the card game Blackjack from scratch. This version of Blackjack is the card aspect only. I have not yet implemented a chip system for betting, nor have I included the ability to split when a pair is drawn. Blackjack is my favorite card game, and it seemed fitting to make, given the requirements for the project, and my prerequisite knowledge of the game's rules. The entire program was created from the ground up, and no reference code was used throughout the entirety of the project.

The player is dealt two cards and one of the dealer cards is shown. The player is then given the choice to "hit" or "stay". If the player chooses to hit, then the player is given another card. If the player's card total (each card is given a number value, and is added up to find the total) hits or exceeds 21, they automatically stay. The dealer then reveals their second card. After the dealer reveals their second card, if their total is less than or equal to sixteen, they draw until their total exceeds sixteen. The card total of the dealer and the player are compared, and whichever total is closer to 21 wins, given that the total does not exceed 21 (if the player or dealer exceeds a total of 21, they lose the game). If the difference from 21 is the same or if both the dealer and player exceed 21 or "bust" then the game results in a draw. If the player chooses to stay, then their total remains, and the dealer continues the same as previously described.

Summary:

Total Lines: 1025

Lines of Comments: 169

Number of Classes: 5

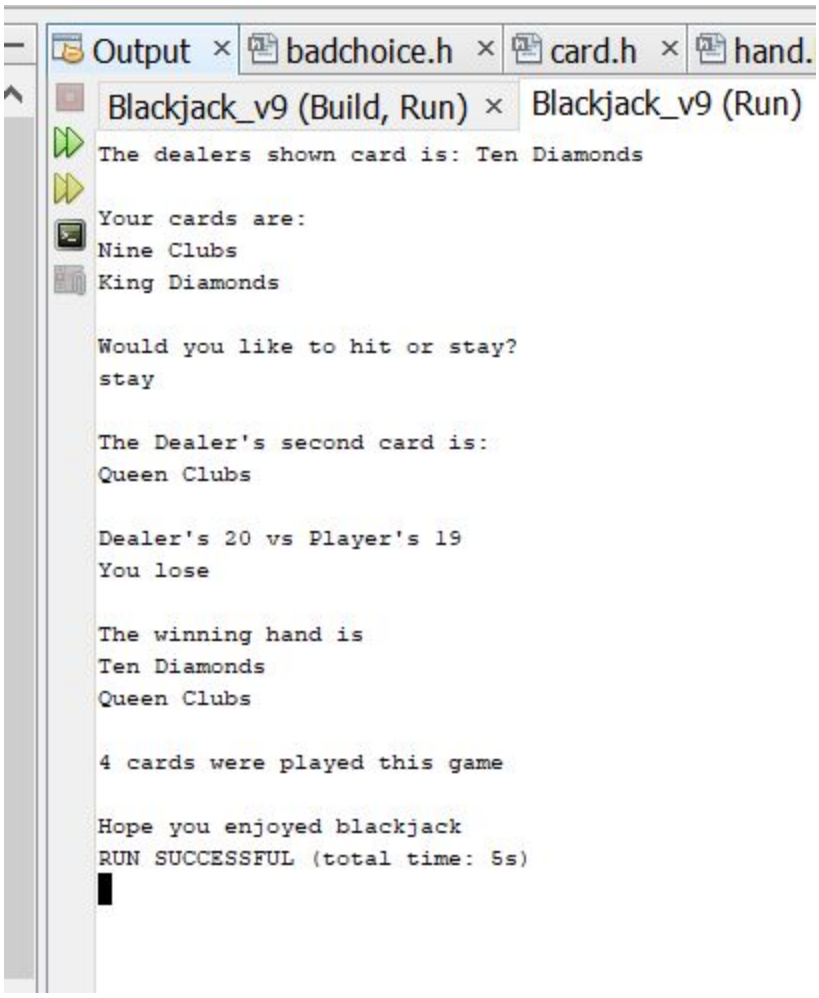
Number of Variables: 24

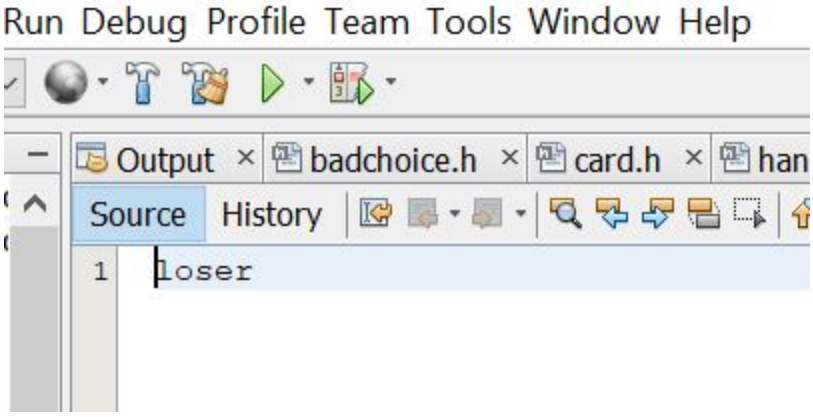
The final project took me about a week to complete. Roughly 20 hours were put into the project including the documentation. The most challenging aspect of the project was implementing all of the required concepts into the program. I could not figure out how to include friends, abstract classes, or templates. Although I functionally understand how each work, implementing them into my project was difficult, and time was limited.

Description:

The game initially started as just a program that drew cards and output them. The card values and the ability to total them were then added, as well as the conditions for winning the game. Next came checks to ensure that repeat cards did not show up. After this came the rest of the concepts from the class up until this point such as pointers and binary files. After this came the splitting of the project into multiplier source files, followed by conversion into classes. After converting into classes, I added exceptions, static variables and utilized a part of the STL. The final version added the copy constructor, and other minor tweaks to the final result.

Sample output:





Output × badchoice.h × card.h × hand.

Blackjack_v9 (Build, Run) × Blackjack_v9 (Run)

```
The dealers shown card is: Six Diamonds

Your cards are:
Seven Clubs
Two Clubs

Would you like to hit or stay?
hit
You drew Seven Diamonds
Would you like to hit or stay?
hit
You drew Nine Diamonds

The Dealer's second card is:
Four Hearts

The dealer draws a card
The next card is
Five Spades

The dealer draws a card
The next card is
Queen Diamonds

Dealer's 25 vs Player's 25
No winner
No winner

The winning hand is
No winning hand
It was a draw

8 cards were played this game

Hope you enjoyed blackjack
RUN SUCCESSFUL (total time: 6s)
```

Output × badchoice.h × card.h ×

Source History

```
1  draw
```

Version 9 Pseudocode

```
Main
{
Set random number seed
Define classes and variables

Draw dealer cards
Check for/replace repeat cards
If an ace is drawn, make its value 11 if it doesn't bust the dealer
Increment number of cards in play

Draw player cards
Check for/replace repeat cards
Increment number of cards in play

Output dealer's first card and player's two cards
If the player draws a(n) ace(s) gets their choice for the aces value(only allows 1 or 11)
Gets the player's choice to hit or stay with input validation

If stay is chosen,
        Calculates player and dealer card total
        Reveals dealer's second card
        If dealer's card total is <= 16,
                draws cards, iterates cards in play, and checks
                for/replaces repeats until a total of >16 is achieved
                If dealer draws ace, sets it to 11 if it doesn't bust the dealer
                Calculates dealer card total
                Output dealer's cards
```

If hit is chosen, loops until stay is chosen

Calculates player and dealer card total
Draw card and iterate cards in play
Checks for/replaces repeat cards

Display card drawn
If ace is drawn, get choice for ace value(only allows 1 or 11)
Add card value to player card total

If player card total >= 21, set choice to stay
If player card total < 21, get choice to hit or stay(with input validation)

Reveals dealer's second card
If dealer's card total is <= 16,

draws cards, iterates cards in play, and checks
for/replaces repeats until a total of >16 is achieved
If dealer draws ace, sets it to 11 if it doesn't bust the dealer
Calculates dealer card total
Output dealer's cards

Find difference of dealer and player total from 21
Find and output game results based on difference

Output the game results

Define character arrays to hold result message and game name
If win result is win, message is winner
If win result is lose, message is loser
If win result is draw, message is draw

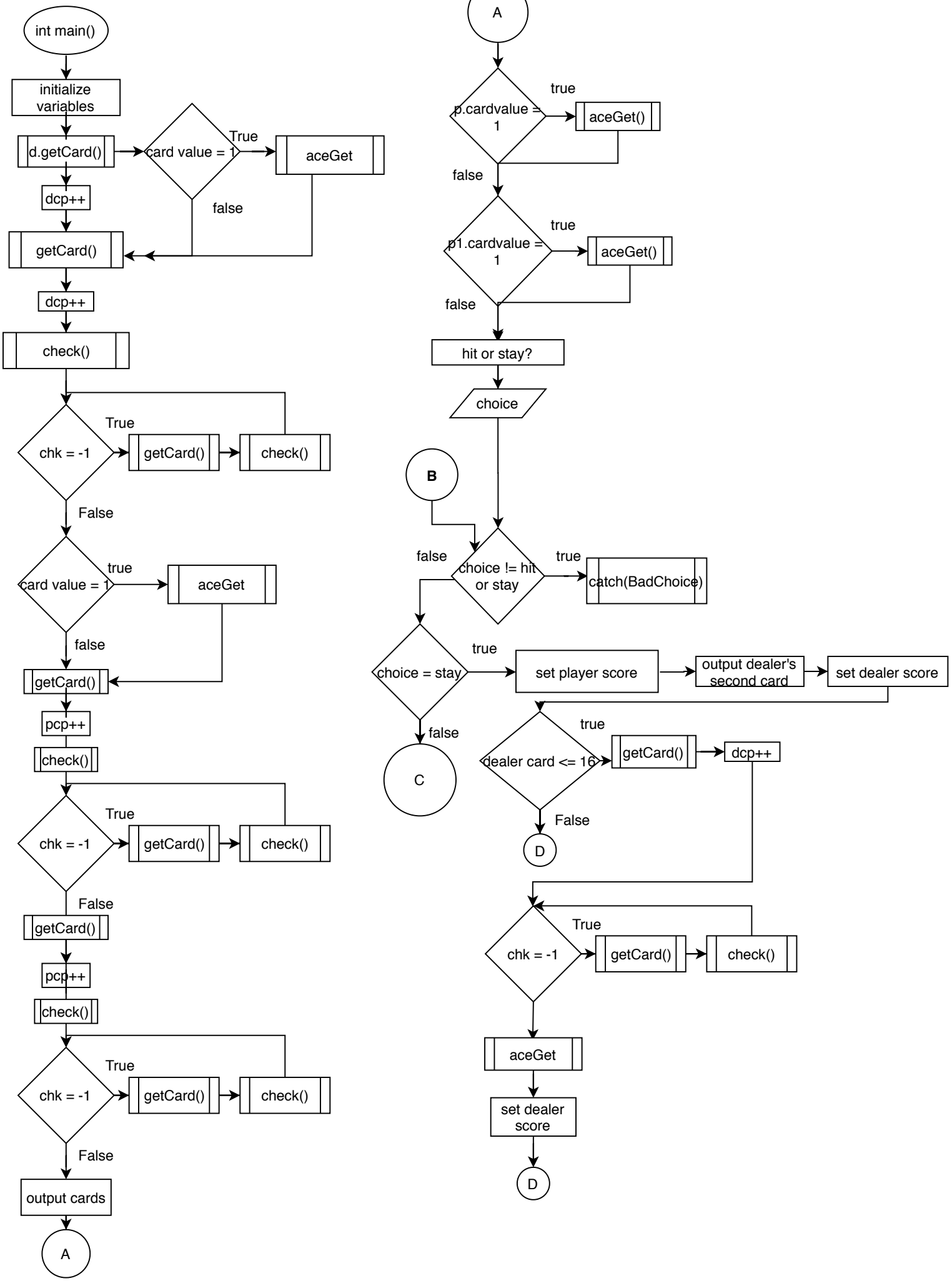
Write the result message to a binary file

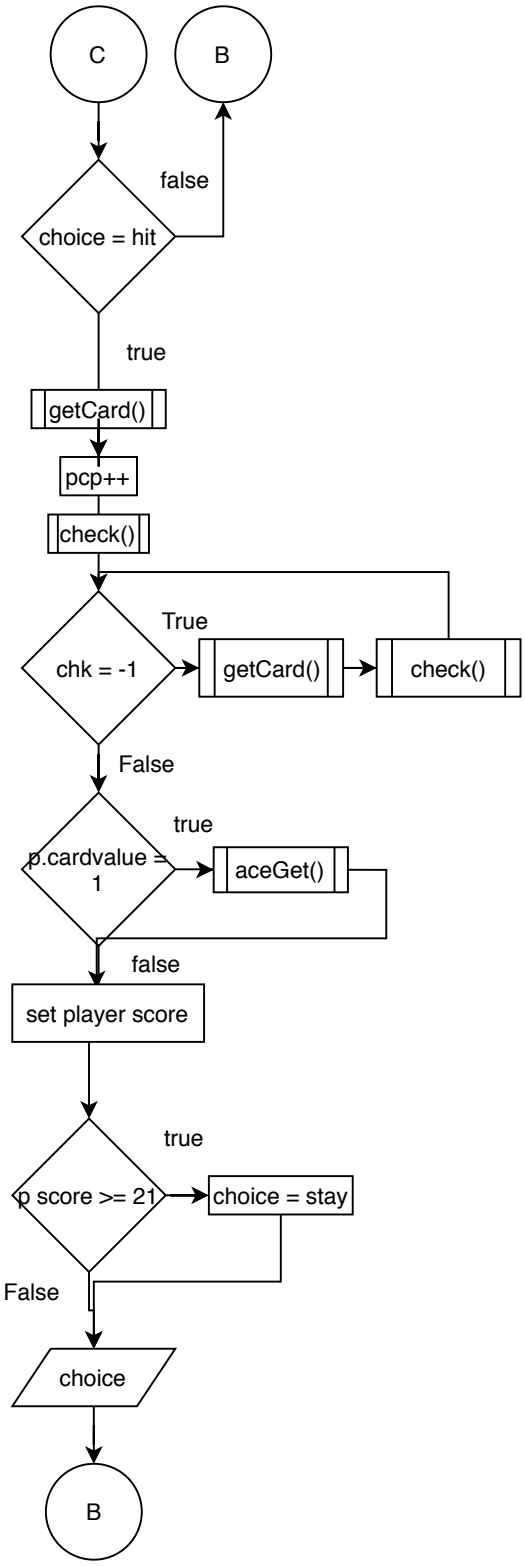
Output the number of cards used in the game
Output exit message

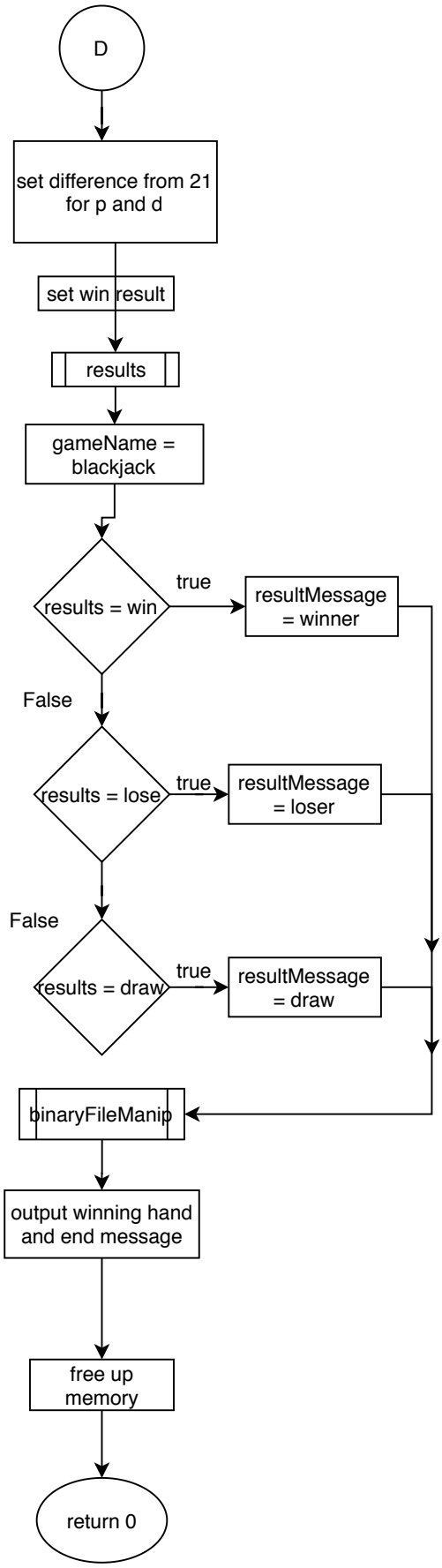
Free up the used memory and exit the program

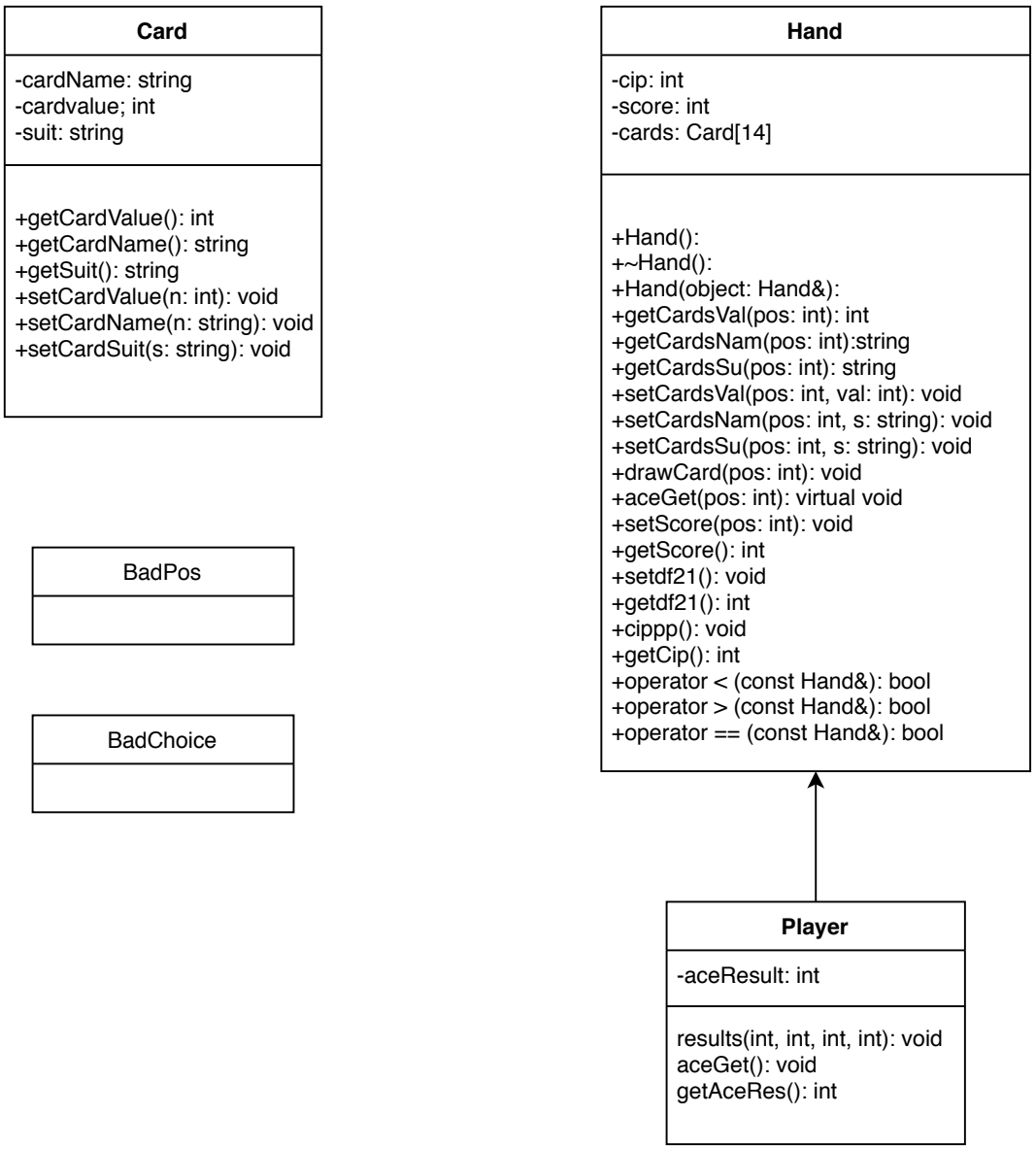
}

Note: I use abbreviations for variable names




C
B
false
choice = hit
true
getCard()
pcp++
check()
True
chk = -1
getCard()
check()
False
true
p.cardvalue = 1
aceGet()
false
set player score
true
p score >= 21
choice = stay
False
choice
B


D
set difference from 21 for p and d
set win result
results
gameName = blackjack
true
results = win
resultMessage = winner
False
true
results = lose
resultMessage = loser
False
true
results = draw
resultMessage = draw
binaryFileManip
output winning hand and end message
free up memory
return 0


Card
-cardName: string
-cardvalue; int
-suit: string
+getCardValue(): int
+getCardName(): string
+getSuit(): string
+setCardValue(n: int): void
+setCardName(n: string): void
+setCardSuit(s: string): void
Hand
-cip: int
-score: int
-cards: Card[14]
+Hand():
+~Hand():
+Hand(object: Hand&):
+getCardsVal(pos: int): int
+getCardsNam(pos: int):string
+getCardsSu(pos: int): string
+setCardsVal(pos: int, val: int): void
+setCardsNam(pos: int, s: string): void
+setCardsSu(pos: int, s: string): void
+drawCard(pos: int): void
+aceGet(pos: int): virtual void
+setScore(pos: int): void
+getScore(): int
+setdf21(): void
+getdf21(): int
+cippp(): void
+getCip(): int
+operator < (const Hand&): bool
+operator > (const Hand&): bool
+operator == (const Hand&): bool
BadPos
BadChoice
Player
-aceResult: int
results(int, int, int, int): void
aceGet(): void
getAceRes(): int

Variable Type	Name	Line	NOTE: LINE IS IN MAIN UNLESS OTHER FILE IS STATED			
int	cardValue	card.h, 12				
	cip	hand.h, 10				
	score	hand.h, 12				
	df21	hand.h, 16				
	aceResult	player.h, 9				
	n	hand.cpp, 12				
	s	hand.cpp, 14				
	chk	43				
	dcp	45				
	pcp	47				
	aceChoice	49				
	ac	player.cpp, 8				
string	cardName	card.h, 11				
	suit	card.h, 13				
	choice	41				
enum	gameResult	16				
char	result	blackjack_implementation.cpp, 51				
	arr	blackjack_implementation.cpp, 109				
	resultMessage	453				
	gameName	456				
	readArray1	462				
	readArray2	466				
bool	res	hand.cpp: 104, 117, 130		NOTE: all 3 instances in hand.cpp		

Cross Reference for Project 2

You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #"s	Pts	Notes
13		Classes			
	1 to 3	Instance of a Class	37	4	NOTE, IF LINE # DOES NOT HAVE A FILE, IT IS IN MAIN
	4	Private Data Members	player.h, line 9	4	Never Public
	5	Specification vs. Implementation	hand.h, line 66	4	.h vs. .cpp files Always split
	6	Inline	hand.h, line 61	4	
	7, 8, 10	Constructors	hand.h, line 19	4	Overloading
	9	Destructors	hand.h, line 20	4	
	12	Arrays of Objects	hand.h, line 14	4	
	16	UML	see UML document	4	
14		More about Classes			
	1	Static	hand.h, line 10	5	
	2	Friends		2	
	4	Copy Constructors	hand.h, line 24	5	
	5	Operator Overloading	hand.h, line 108	8	Overload 3 operators
	7	Aggregation	hand.h, line 14	6	
15		Inheritance			
	1	Protected members	hand.h, line 8	6	
	2 to 5	Base Class to Derived	player.h, line 6	6	
	6	Polymorphic associations	hand.h, line 69	6	
	7	Abstract Classes		6	
16		Advanced Classes			
	1	Exceptions	80	6	
	2 to 4	Templates		6	
	5	STL	457	6	
		Sum		100	