Project 2

Title

Blackjack

Course

CIS-17A

Section

42474

Due Date

6/1/2018

Author

Seth Tyler

Table of Contents

Introduction

Blackjack is a famous card-based game found in casinos and other gambling environments. It is one of the simplest games to grasp, which was ultimately the motivation for this program.

This game is completely text-based. While a version with GUI's would be more appealing and enticing to play, this game captures the gameplay sufficiently, while leaving out more complex features of the game such as doubling, splitting, and insurance.

Summary

This program is 608 lines long total. This includes header files and source files.

The program includes 15 functions, 6 system libraries, 4 class files, and 18 variables (not including for-loop initializers.)

Overall, the game was more challenging to create than expected, even without the more complex features mentioned previously. The most challenging aspects were fulfilling the checklist and ensuring every chapter section was included in the code. While not everything was fulfilled, the game works beautifully and uses what is necessary.

In total, time spent on this project was roughly 10 hours. Most of this time was dedicated to fixing bugs and polishing the game, as well as catching up on knowledge that was not as fluent. The line limit of 1000 was unreachable for this particular project without making unnecessary clutter.

Objective

The objective of the game is to make the sum of the cards in your hand as close as possible to 21 without exceeding 21. Face cards are worth 10, aces are worth 11 or 1 (depending on the player's needs.)

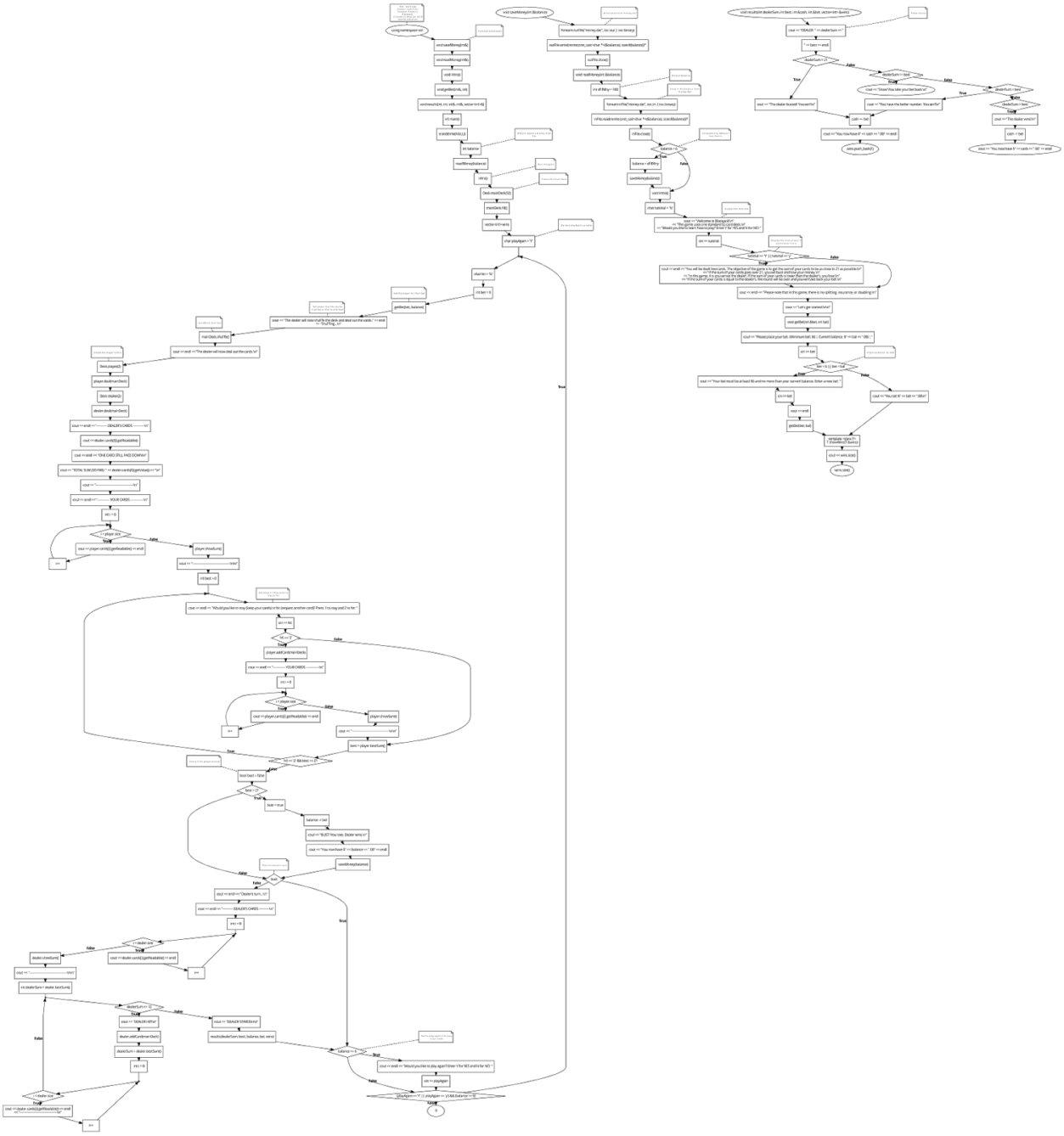
The game begins by placing your bet. The dealer will then deal you two cards face up and two cards to themselves, with one face down and one face up. You must determine if you would like another card (hit) or if you are comfortable with the amount that you have (stay)

If the dealer has a greater sum than you, or you exceed 21, they will win. If the dealer exceeds 21 (busts) or has a lower amount than you, you will win.

Sample Input/Output

```
Welcome to Blackjack!
This game uses one standard 52 card deck.
Would you like to learn how to play? Enter Y for YES and N for NO: y

You will be dealt two cards. The objective of the game is to get the sum of your cards to be as close to 21 as possible.
If the sum of your cards goes over 21, you will bust and lose your money.
In this game, it is you versus the dealer. If the sum of your cards is lower than the dealer's, you lose.
If the sum of your cards is equal to the dealer's, the round will be over and you will take back your bet.

Please note that in this game, there is no splitting, insurance, or doubling.
Let's get started.

Please place your bet. (Minimum bet: $6 | Current balance: $110.00) : 10
You bet $10.00
The dealer will now shuffle the deck and deal out the cards.
Shuffling...

The dealer will now deal out the cards.

---------- DEALER'S CARDS -----------
10 OF HEARTS (VALUE: 10)
ONE CARD STILL FACE DOWN
TOTAL SUM (SO FAR): 10
--------------------------------------

------------ YOUR CARDS -------------
4 OF CLUBS (VALUE: 4)
7 OF SPADES (VALUE: 7)
TOTAL SUM: 11
BEST SUM: 11
--------------------------------------


Would you like to stay (keep your cards) or hit (request another card)? Press 1 to stay and 2 to hit: 2

------------ YOUR CARDS -------------
4 OF CLUBS (VALUE: 4)
7 OF SPADES (VALUE: 7)
10 OF SPADES (VALUE: 10)
TOTAL SUM: 21
BEST SUM: 21
--------------------------------------


Would you like to stay (keep your cards) or hit (request another card)? Press 1 to stay and 2 to hit: 1

Dealer's turn...

---------- DEALER'S CARDS ----------
10 OF HEARTS (VALUE: 10)
8 OF SPADES (VALUE: 8)
TOTAL SUM: 18
BEST SUM: 18
--------------------------------------

DEALER STAYED

DEALER: 18; YOU: 21
You have the better number. You win!
You now have $120.00

Would you like to play again? Enter Y for YES and N for NO: n

RUN SUCCESSFUL (total time: 19s)
```

Flowchart



Variables

Variable	Description	Location
balance	Contain's the player's cash	43
dataFile	fstream to read in player's cash from money.txt	45
playAgain	Determines whether the player wants to play another round	55
hit	Determines if the player wants to hit or stay	63
bet	Contains the player's bet amount	71
hand	A Deck variable for the player's hand	84
dealer	A Deck variable for the dealer's hand	88
best	Contains the best sum of the player's hand	92
bust	Flag variable for if the player's sum exceeded 21. Round is over and player loses their money if true	130
tutorial	Determines if the player wants to know how to play	172
cardNum	An iterator for each of the 52 cards to be added into the deck	193
sum	The first sum of cards in a hand	298/326
aceSum	The second sum of cards in a hand that is shown if there is an ace in a hand	299/327
numAces	The number of aces in a hand	300/328

Checklist

Chapter	Section	Topic	Where Line #"s	Pts	Notes
13		Classes			
	1 to 3	Instance of a Class	Deck.h, Card.h, NormalCard.h, SpecialCard.h	4	
	4	Private Data Members	Card.h	4	Never Public
	5	Specification vs. Implementation	Card.h, Card.cpp	4	.h vs. .cpp files Always split
	6	Inline	Card.h	4	
	7, 8, 10	Constructors	Deck.h	4	Overloading
	9	Destructors	Deck.h	4	
	12	Arrays of Objects	Deck.h	4	
	16	UML		4	
14		More about Classes			
	1	Static	Deck.h	5	
	2	Friends		2	
	4	Copy Constructors	Deck.cpp, Line 77	5	
	5	Operator Overloading	Deck.h, Card.h, Deck.cpp, Card.cpp	8	Overload 3 operators
	7	Aggregation	Deck.h	6	
15		Inheritance	NormalCard.h, SpecialCard.h		
	1	Protected members	Card.h	6	
	2 to 5	Base Class to Derived	NormalCard.h, SpecialCard.h	6	
	6	Polymorphic	Deck.cpp	6	

		associations			
	7	Abstract Classes	Deck.h	6	
16		Advanced Classes			
	1	Exceptions		6	
	2 to 4	Templates	main.cpp Line 210	6	
	5	STL	main.cpp Line 44	6	
		Sum		100	

main.cpp

```
/*
```

```
 * File:   main.cpp
 * Author: Seth Tyler
 * Purpose: Project 2: Blackjack
 * Created on May 28, 2018
 */

// System Libraries
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <ctime>
#include <vector>
#include "Card.h"
#include "NormalCard.h"
#include "SpecialCard.h"
#include "Deck.h"

using namespace std;

// Function prototypes
void saveMoney(int&);
void readMoney(int&);
void intro();
void getBet(int&, int);
void results(int, int, int&, int&, vector<int>&);

int main() {

        srand(time(NULL));

        // Read in player's money from file
        int balance;
        readMoney(balance);

        // Run the game
        intro();

        // Create the main deck
        Deck mainDeck(52);
        mainDeck.fill();

        vector<int> wins;

        // Declare playAgain variable
        char playAgain = 'Y';

        do {

        char hit = 'N';
        int bet = 0;

        // Ask the player for their bet
        getBet(bet, balance);

        // Tell player that the dealer is going to shuffle and deal
        cout << "The dealer will now shuffle the deck and deal out the cards." << endl
```

```
                << "Shuffling...\n";
        //mainDeck.shuffle();
        mainDeck.shuffle();

        cout << endl <<"The dealer will now deal out the cards.\n";

        // Create the player's deck
        Deck player(2);
        player.deal(mainDeck);

        Deck dealer(2);
        dealer.deal(mainDeck);

        cout << endl << "---------- DEALER'S CARDS -----------\n";
        cout << dealer.cards[0].getReadable();
        cout << endl << "ONE CARD STILL FACE DOWN\n";
        cout << "TOTAL SUM (SO FAR): " << dealer.cards[0].getValue() << "\n";
        cout << "-------------------------------------\n";

        cout << endl << "------------ YOUR CARDS -------------\n";
        for (int i = 0; i < player.size; i++) {
        cout << player.cards[i].getReadable() << endl;
        }
        player.showSum();
        cout << "-------------------------------------\n\n";

        int best = 0;

        // Ask player if they want to stay or hit
        do {
        cout << endl << "Would you like to stay (keep your cards) or hit (request another card)? Press 1
to stay and 2 to hit: ";
        cin >> hit;

        if (hit == '2') {
                player.addCard(mainDeck);
                cout << endl << "------------ YOUR CARDS -------------\n";
                for (int i = 0; i < player.size; i++) {
                cout << player.cards[i].getReadable() << endl;
                }
                player.showSum();
                cout << "-------------------------------------\n\n";
        }

        best = player.bestSum();
        } while (hit == '2' && best <= 21);

        // Check if the player busted
        bool bust = false;
        if (best > 21) {
        bust = true;
        balance -= bet;
        cout << "BUST! You lose. Dealer wins.\n";
        cout << "You now have $" << balance << ".00" << endl;
        saveMoney(balance);
        }
```

```
        // Play the dealer's turn
        if (!bust) {
        cout << endl << "Dealer's turn...\n";

        cout << endl << "---------- DEALER'S CARDS ----------\n";
        for (int i = 0; i < dealer.size; i++) {
                cout << dealer.cards[i].getReadable() << endl;
        }
        dealer.showSum();
        cout << "-------------------------------------\n\n";

        int dealerSum = dealer.bestSum();
        while (dealerSum <= 12) {
                cout << "DEALER HIT!\n";
                dealer.addCard(mainDeck);
                dealerSum = dealer.bestSum();
                for (int i = 0; i < dealer.size; i++) {
                cout << dealer.cards[i].getReadable() << endl
                        << "-------------------------------------\n";
                }
        }
        cout << "DEALER STAYED\n\n";

        results(dealerSum, best, balance, bet, wins);
        }

        // Ask to play again if money is sufficient
        if (balance >= 6) {
        cout << endl << "Would you like to play again? Enter Y for YES and N for NO: ";
        cin >> playAgain;
        }

        } while ((playAgain == 'Y' || playAgain == 'y') && (balance >= 6));

        return 0;
}

void saveMoney(int &balance) {
        // Write balance to money.dat
        fstream outFile("money.dat", ios::out | ios::binary);
        outFile.write(reinterpret_cast<char *>(&balance), sizeof(balance));
        outFile.close();
}

void readMoney(int &balance) {
        // Default balance
        int dfltMny = 100;

        // Read in the balance from money.dat
        fstream inFile("money.dat", ios::in | ios::binary);
        inFile.read(reinterpret_cast<char *>(&balance), sizeof(balance));
        inFile.close();

        // Set balance to default if less than 6
        if (balance < 6) {
        balance = dfltMny;
        saveMoney(balance);
```

```
        }
}

void intro() {
        char tutorial = 'N';

        // Output the intro text
        cout << "Welcome to Blackjack!\n"
        << "This game uses one standard 52 card deck.\n"
        << "Would you like to learn how to play? Enter Y for YES and N for NO: ";
        cin >> tutorial;

        // Display the instructions if user inputs Y or y
        if (tutorial == 'Y' || tutorial == 'y') {
        cout << endl << "You will be dealt two cards. The objective of the game is to get the sum of your
cards to be as close to 21 as possible.\n"
                << "If the sum of your cards goes over 21, you will bust and lose your money.\n"
                << "In this game, it is you versus the dealer. If the sum of your cards is lower than the
dealer's, you lose.\n"
                << "If the sum of your cards is equal to the dealer's, the round will be over and you will
take back your bet.\n";
        }

        cout << endl << "Please note that in this game, there is no splitting, insurance, or doubling.\n";
        cout << "Let's get started.\n\n";
}

void getBet(int &bet, int bal) {
        cout << "Please place your bet. (Minimum bet: $6 | Current balance: $" << bal << ".00) : ";
        cin >> bet;
        // Input validation for bet
        if (bet < 6 || bet > bal) {
        cout << "Your bet must be at least $6 and no more than your current balance. Enter a new bet: ";
        cin >> bet;
        cout << endl;
        getBet(bet, bal);
        } else {
        cout << "You bet $" << bet << ".00\n";
        }
}

template <class T>
T showWins(T &wins) {
        cout << wins.size();
        return wins.size();
}

void results(int dealerSum, int best, int &cash, int &bet, vector<int> &wins) {
        // Show results
        cout << "DEALER: " << dealerSum << "; YOU: " << best << endl;
        if (dealerSum > 21) {
        cout << "The dealer busted! You win!\n";
        cash += bet;
        cout << "You now have $" << cash << ".00" << endl;
        wins.push_back(1);
        } else if (dealerSum == best) {
        cout << "Draw! You take your bet back.\n";
```

```
        } else if (dealerSum < best) {
        cout << "You have the better number. You win!\n";
        cash += bet;
        cout << "You now have $" << cash << ".00" << endl;
        wins.push_back(1);
        } else if (dealerSum > best) {
        cout << "The dealer wins!\n";
        cash -= bet;
        cout << "You now have $" << cash << ".00" << endl;
        }
}
```