**BlackJack Slot Machine**

**California State University, Long Beach**

**By Taylor Bui**

**& (Separated midway)**

**Marsilinou Zaky**

**December 12th 2018**

Introduction

This program simulates a BlackJack game. First the program will ask the user for an account Number and money since the balance is set to 0. A deck is created and two hands will be created, a player hand and a dealer hand. The player hand is revealed while one of the dealer cards is face down. The player is allowed to “Hit”, “Stand” or “Split.” “Hit” is when the player wants to draw a card. “Stand is when the player wants to end their turn. “Split” is when the player wants to divide their hand into two separate hands. After the player turn the dealer turn automatically runs and compares who wins or loses. The user can choose to play again or not.

Program Analysis and algorithm design

The variable used in this program are rank, suit, facedUp, dealerCards, playerCards, data, playerHandSize, dealHandSize, decksize, betAmount, accountNumber, balance, splitter, and earning. Rank and suit are characters that describe which card is being picked up. facedUp is a Boolean variable that is true when card is revealed and false when card is faced down. dealerCards and playerCards are vectors of the class Card that hold the hands of the dealer and the player. data is a vector of cards as well that hold the remainder of the deck that is not in play. playerHandSize, dealHandSize, deckSize are variables that simplify getting the size of the vectors. betAmount is a double type that contains the amount the player inputted to bet. accountNumber stores the account number of the user. Splitter is a Boolean statement when false the player did not choose to split, when true then the player has chosen to split. Earning is a double type that stores the total net loss of the player.

There are many functions used in this program, I will skip all of the getter and setter functions. Starting with Deck.h, the display() function shows all cards either in the deck or the hand of the player or dealer. shuffle() randomizes the order of the deck. sumHand(vector<Card> vect) counts the total of the value of the vector passed into it. dealToPlayer(Card c) and dealtoDealer(card c) increases the hand size of the user, decreases deck size, and puts the next deck card into the user’s hand. dealerTurn() is the method that decides what the dealer will do. If the dealer card values are 15 or less then the dealer will hit for a hard. reset() deletes the vector of both player and dealer hands and it also reduces their hand size to 0 in order to play the game again. hit() is the function called to draw a card. The last class is Game. Game just has a Menu() that is the game menu and calls everything else.

The two main algorithms are creating the deck or the vector of Cards, data and the shuffle. The deck is constructed using suits and cards then they are pushed onto a vector. The shuffle randomizes the data vector.

Program Code

Main.cpp

//

// main.cpp

// BlackJack

//

// Created by Taylor Bui on 12/11/18.

// Copyright (c) 2018 Taylor Bui. All rights reserved.

//

#include <iostream>

#include "Deck.h"

#include "Game.h"

#include "Player.h"

#include "Card.h"

using namespace std;

int main(int argc, const char \* argv[])

{

Game g;

g.Menu();

}

Game.h

//

// Game.h

// blackjack

//

// Created by Taylor Bui on 12/10/18.

// Copyright (c) 2018 Taylor Bui. All rights reserved.

//

#ifndef \_\_blackjack\_\_Game\_\_

#define \_\_blackjack\_\_Game\_\_

#include <iostream>

using namespace std;

class Game

{

public:

Game();

void Menu();

};

#endif

Game.cpp

//

// Game.cpp

// blackjack

//

// Created by Taylor Bui on 12/10/18.

// Copyright (c) 2018 Taylor Bui. All rights reserved.

//

#include "Game.h"

#include "Player.h"

#include "Deck.h"

#include <string.h>

Game::Game ()

{}

void Game:: Menu()

{

double accountNum;

double bet = 0;

int input = 0;

char playAgain ;

bool repeat = true;

double money;

int counter= 0;

cout << "Welcome to blackjack program by Taylor Bui and Marsilinou Zaky" << endl;

cout << "Enter account number: ";

cin >> accountNum;

Player p (accountNum);

Deck d;

d.shuffle();

cout << endl;

do

{

cout << "Your account balance is currently: $" << p.getBalance() << endl;

if(p.getBalance() == 0)

{

cout << "Input how much money you want to insert: $";

cin >> money;

money = p.getBalance() + money;

p.setBalance(money);

cout << "New balance is: $" << p.getBalance() << endl;

}

cout << "Enter how much you want to bet: $" ;

cin >> bet;

p.setBetAmount(bet);

while(bet > p.getBalance())

{

cout << "You do not have enough money. Your balance is: $" << p.getBalance()<<". Enter how much you want to bet: $" ;

cin >> bet;

p.setBetAmount(bet);

}

cout << endl << "Blackjack started: " << endl;

//Player card 1 draw + flip up

d.dealToPlayer(d.deal());

d.getPlayerCards()[d.getPlayerCards().size()-1].faceUp();

//Dealer card 1 draw + flip up

d.dealToDealer(d.deal());

d.getDealerCards()[d.getDealerCards().size()-1].faceUp();

//Player card 2 draw + flip up

d.dealToPlayer(d.deal());

d.getPlayerCards()[d.getPlayerCards().size()-1].faceUp();

//Daeler card 2 draw + left unflipped

d.dealToDealer(d.deal());

//Display player cards

cout << "Your cards: ";

for(int i = 1; i< d.getPlayerCards().size()+1; i++)

{

d.getPlayerCards()[d.getPlayerCards().size()-i].display();

}

//display value of hand

cout << " = " << d.sumHand(d.getPlayerCards()) ;

cout << endl;

//display all dealer cards

cout << "Dealer cards: ";

for(int i = 1; i< d.getDealerCards().size()+1; i++)

{

d.getDealerCards()[d.getDealerCards().size()-i].display();

}

cout << endl;

//Hit stand or split

do

{

cout << "Do you want to 1)hit, 2)stand, or 3)split : ";

cin >>input;

switch(input)

{

case 1: d.hit();

break;

case 2: p.stand();

break;

case 3: p.split(d);

break;

}

}while(input != 2);

//flip up first dealer card

d.getDealerCards()[d.getDealerCards().size()-1].faceUp();

d.dealerTurn();

//display all dealer cards

cout << "Dealer cards: ";

for(int i = 1; i< d.getDealerCards().size()+1; i++)

{

d.getDealerCards()[d.getDealerCards().size()-i].display();

}

cout << " = " << d.sumHand(d.getDealerCards()) ;

cout << endl;

p.result(d);

cout << endl <<"Would you like to play again Y/N? : ";

cin >> playAgain;

playAgain = tolower(playAgain);

if( playAgain == 'n')

{

repeat = false;

}

counter++;

if(counter == 5)

{

d.shuffle();

counter = 0;

}

d.reset();

}while(repeat == true);

cout << "Thank you for playing" << endl;

if(p.getBalance() > 0)

{

cout << "Your balance is: $" << p.getBalance() <<". Go to the cashier to get your money. " <<endl;

}

cout << "Your total earning is: $" << p.getEarning();

}

Player.h

//

// player.h

// final 282

//

// Created by Taylor Bui on 12/3/18.

// Copyright (c) 2018 Taylor Bui. All rights reserved.

//

#ifndef \_\_final\_282\_\_player\_\_

#define \_\_final\_282\_\_player\_\_

#include "Deck.h"

#include <iostream>

using namespace std;

class Player {

public:

Player();

Player (double betAmount, double accountNumber, double balance);

Player(double accountNumber);

Player (double betAmount, double accountNumber, double balance, bool splitter);

// void hit(Deck d) ;

void stand();

void split(Deck d);

void result(Deck d);

void setBetAmount(double bet);

double getBalance();

void setBalance(double money);

double getEarning();

private:

double betAmount ;

double accountNumber ;

double balance;

bool splitter;// if this object was created by split then do this instead

double earning;

};

#endif /\* defined(\_\_final\_282\_\_player\_\_) \*/

Player.cpp

//

// player.cpp

// final 282

//

// Created by Taylor Bui on 12/3/18.

// Copyright (c) 2018 Taylor Bui. All rights reserved.

//

#include "Player.h"

#include "Deck.h"

Player :: Player()

{

betAmount = 0;

accountNumber = 0;

balance = 0;

splitter = false;

earning = 0;

}

Player :: Player(double accountNumber)

{

(\*this).accountNumber = accountNumber;

betAmount = 0;

balance= 0;

splitter = false;

earning = 0;

}

Player::Player( double betAmount, double accountNumber, double balance)

{

(\*this).betAmount = betAmount;

(\*this).accountNumber = accountNumber;

(\*this).balance = balance;

splitter = false;

earning = 0;

}

void Player::setBetAmount(double bet)

{

betAmount = bet;

}

Player::Player( double betAmount, double accountNumber, double balance, bool splitter)

{

(\*this).betAmount = betAmount;

(\*this).accountNumber = accountNumber;

(\*this).balance = balance;

splitter = true;

}

//Creates a new object for a new hand. Also checks if there is a duplicate

void Player:: split(Deck d)

{

//checks if its possible to split.

int duplicateCheck = 0;

duplicateCheck = d.sumHand(d.getPlayerCards())%2;

if(balance >= betAmount\*2 && duplicateCheck == 0 && d.getPlayerCards().size() == 2 )

{

splitter = true;

balance = betAmount + betAmount;

}else

cout << " Cannot split" << endl;

}

//print statement

void Player:: stand()

{

cout << "You have picked to stand" << endl;

}

//function to call to get the result of the game

void Player :: result(Deck d)

{

cout<< "Player total :"<<d.sumHand(d.getPlayerCards()) <<". Dealer total: " <<d.sumHand(d.getDealerCards())<< endl;

//player loses

if(d.sumHand(d.getPlayerCards()) >21 && d.sumHand(d.getPlayerCards()) < d.sumHand(d.getDealerCards()) && d.sumHand(d.getDealerCards()) <22 )

{

balance = balance - betAmount;

cout << "You lost $" << betAmount<< ". Your new balance is " << balance << endl;

earning = earning-betAmount;

}else

if( d.sumHand(d.getPlayerCards()) == d.sumHand(d.getDealerCards()) )

{

//tie

balance = balance - (betAmount/2);

cout << "You tied. You lost $" << .5\*betAmount<< ". Your new balance is " << balance << endl;

earning = earning - (betAmount/2);

}else

if(d.sumHand(d.getPlayerCards()) >21 )

{

//lose above 21

balance = balance - betAmount;

cout << "You lost $" << betAmount<< ". Your new balance is $" << balance << endl;

earning = earning -betAmount;

}

else

if(d.sumHand(d.getPlayerCards()) < d.sumHand(d.getDealerCards()) && d.sumHand(d.getDealerCards()) <= 21 )

{

//lose

balance = balance - betAmount;

cout << "You lost $" << betAmount<< ". Your new balance is $" << balance << endl;

earning = earning -betAmount;

}else

{

//win

balance = balance + betAmount;

cout<< "You won $" << betAmount <<" .Your new balance is $" << balance << endl;

earning = earning + betAmount;

}

betAmount = 0;

}

double Player::getBalance()

{

return balance;

}

void Player::setBalance(double money)

{

balance = money;

}

double Player::getEarning()

{

return earning;

}

Deck.h

#ifndef DECK\_H

#define DECK\_H

#include <string>

#include <iostream>

#include "Card.h" //fix to my problem it was a cpp

using namespace std;

#include <vector>

class Deck{

public:

Deck();

void display();

void shuffle();

Card deal();

bool isEmpty();

int sumHand(vector<Card> c);

vector<Card>& getDealerCards();

vector<Card>& getPlayerCards();

void dealToPlayer(Card c);

void dealToDealer(Card c);

int getPlayerHandSize() const;

int getDealHandSize() const;

int getDeckSize() const;

void dealerTurn();

void reset();

void hit() ;

friend class Player;

private:

vector<Card> dealerCards;

vector<Card> playerCards;

vector<Card> data;

int playerHandSize;

int dealHandSize;

int deckSize;

};

#endif

Deck.cpp

#include "Deck.h"

#include <iostream>

#include <ctime>

#include <cstdlib>

using namespace std;

vector<Card> dealerCards;

vector<Card> playerCards;

Deck::Deck(){

playerHandSize = 0;

dealHandSize = 0;

deckSize = 52;

char suits[] = {'H','D','C','S'};

char cards[] = {'2','3','4','5','6','7','8','9','T','J','Q','K','A'};

for (int j=0; j<13; j++)

for (int i=0; i<4; i++)

data.push\_back(Card(cards[j],suits[i], false));

}

void Deck::shuffle(){

int randomIndex;

int randomNum = (rand() % 27);

srand(time(0));

for (int c = 0; c < 200; c++) {

for (int i = 0; i < 52; i++) {

randomIndex = (rand() % 52);

Card temp = data[randomIndex];

data[randomIndex] = data[i];

data[i] = temp;

}

}

}

void Deck::display() {

int c = 0;

cout << "\n\t\t";

for (int counter = 0; counter < 52; counter++){

if (c == 13) {

cout << "\n\t\t";

data[counter].display();

c = 1;

} else {

data[counter].display();

c++;

}

}

}

bool Deck::isEmpty(){

if (deckSize > 0) return 0;

return 1;

}

int Deck::sumHand(vector<Card> c){

int sum = 0;

int aces = 0;

for (int i = 0; i<c.size() ; i++){

if ( c[i].getRank() != 'A')

sum += c[i].getValue();

else aces++;

}

for (int i =1;i<aces;i++)

sum++;{

if (aces > 0){

if(sum + 11 <= 21)

sum+=11;

else sum++;

}

}

return sum;

}

Card Deck::deal(){

Card c;

if(! this->isEmpty()){

c = data[data.size()-1];

data.pop\_back();

return c;

}

else return c;

}

void Deck::dealToPlayer(Card c){

playerCards.push\_back(c);

playerHandSize++;

deckSize--;

}

void Deck::dealToDealer(Card c){

dealerCards.push\_back(c);

dealHandSize++;

deckSize--;

}

vector<Card>& Deck::getDealerCards(){

return dealerCards;

}

vector<Card>& Deck::getPlayerCards(){

return playerCards;

}

int Deck::getPlayerHandSize() const{

return playerHandSize;

}

int Deck::getDealHandSize() const{

return dealHandSize;

}

int Deck::getDeckSize() const{

return deckSize;

}

void Deck::dealerTurn()

{

cout << endl;

while(sumHand(getDealerCards()) <= 15)

{

cout << "Dealer drew: ";

getDealerCards()[getDealerCards().size()-1].faceUp();

dealToDealer(deal());

getDealerCards()[getDealerCards().size()-1].faceUp();

getDealerCards()[getDealerCards().size()-1].display();

}

cout << endl;

}

void Deck::reset()

{

dealerCards.clear();

playerCards.clear();

playerHandSize = 0;

dealHandSize = 0;

}

//draw a card/deal

void Deck::hit()

{

cout << endl << "Drawing a card: " << endl;

dealToPlayer(deal());

getPlayerCards()[getPlayerHandSize()-1].faceUp();

cout << "You drew: ";

getPlayerCards()[getPlayerHandSize()-1].display();

cout << endl;

cout<<"Your hand is: ";

for(int i = 1; i< getPlayerHandSize()+1 ; i++)

{

getPlayerCards()[getPlayerHandSize()-i].display();

}

cout << " = " << sumHand(getPlayerCards()) ;

cout << endl;

}

Card.h

#ifndef CARD\_H

#define CARD\_H

#include <string>

#include <iostream>

using namespace std;

class Card

{

public:

Card();

Card(char rnk, char st, bool fup);

char getRank() const;

char getSuit() const;

void faceUp();

void display();

int getValue();

private:

char rank;

char suit;

bool facedUp;

};

#endif

Card.cpp

#include "Card.h"

#include <iostream>

using namespace std;

Card::Card(){

rank = '\0';

suit = '\0';

facedUp = false;

}

Card::Card(char rnk, char st, bool fup){

rank = rnk;

suit = st;

facedUp = fup;

}

void Card:: display(){

if(!facedUp){

cout << "|\* \*|";

return;

}

if (rank == 'T') cout << "|10" << suit << "|";

else cout << "|" << (\*this).rank << " " << (\*this).suit << "|";

}

int Card::getValue(){

if (rank == 'A') return 10;

else if (rank == 'T' || rank == 'K' || rank == 'Q' || rank =='J')return 10;

else return rank-'0';

}

char Card::getRank() const{

return rank;

}

char Card::getSuit() const{

return suit;

}

void Card::faceUp(){

facedUp = true;

}

**Sample Run**

**Welcome to blackjack program by Taylor Bui and Marsilinou Zaky**

**Enter account number:** 123

**Your account balance is currently: $0**

**Input how much money you want to insert: $**1000

**New balance is: $1000**

**Enter how much you want to bet: $**100

**Blackjack started:**

**Your cards: |10C||8 S| = 18**

**Dealer cards: |\* \*||9 D|**

**Do you want to 1)hit, 2)stand, or 3)split :** 3

**Cannot split**

**Do you want to 1)hit, 2)stand, or 3)split :** 2

**You have picked to stand**

**Dealer drew: |9 H|**

**Dealer cards: |9 H||5 H||9 D| = 23**

**Player total :18. Dealer total: 23**

**You won $100 .Your new balance is $1100**

**Would you like to play again Y/N? :** y

**Your account balance is currently: $1100**

**Enter how much you want to bet: $**200

**Blackjack started:**

**Your cards: |K H||8 C| = 18**

**Dealer cards: |\* \*||Q C|**

**Do you want to 1)hit, 2)stand, or 3)split :** 1

**Drawing a card:**

**You drew: |7 S|**

**Your hand is: |7 S||K H||8 C| = 25**

**Do you want to 1)hit, 2)stand, or 3)split :** 2

**You have picked to stand**

**Dealer cards: |10D||Q C| = 20**

**Player total :25. Dealer total: 20**

**You lost $200. Your new balance is $900**

**Would you like to play again Y/N? :** y

**Your account balance is currently: $900**

**Enter how much you want to bet: $**200

**Blackjack started:**

**Your cards: |3 D||5 S| = 8**

**Dealer cards: |\* \*||7 D|**

**Do you want to 1)hit, 2)stand, or 3)split :** 1

**Drawing a card:**

**You drew: |2 H|**

**Your hand is: |2 H||3 D||5 S| = 10**

**Do you want to 1)hit, 2)stand, or 3)split :** 1

**Drawing a card:**

**You drew: |2 C|**

**Your hand is: |2 C||2 H||3 D||5 S| = 12**

**Do you want to 1)hit, 2)stand, or 3)split :** 1

**Drawing a card:**

**You drew: |9 S|**

**Your hand is: |9 S||2 C||2 H||3 D||5 S| = 21**

**Do you want to 1)hit, 2)stand, or 3)split :** 2

**You have picked to stand**

**Dealer cards: |J S||7 D| = 17**

**Player total :21. Dealer total: 17**

**You won $200 .Your new balance is $1100**

**Would you like to play again Y/N? :** y

**Your account balance is currently: $1100**

**Enter how much you want to bet: $**2000

**You do not have enough money. Your balance is: $1100. Enter how much you want to bet: $**100

**Blackjack started:**

**Your cards: |7 C||6 D| = 13**

**Dealer cards: |\* \*||J C|**

**Do you want to 1)hit, 2)stand, or 3)split :** 1

**Drawing a card:**

**You drew: |3 C|**

**Your hand is: |3 C||7 C||6 D| = 16**

**Do you want to 1)hit, 2)stand, or 3)split :** 2

**You have picked to stand**

**Dealer drew: |4 S|**

**Dealer cards: |4 S||3 S||J C| = 17**

**Player total :16. Dealer total: 17**

**You lost $100. Your new balance is $1000**

**Would you like to play again Y/N? :** y

**Your account balance is currently: $1000**

**Enter how much you want to bet: $**200

**Blackjack started:**

**Your cards: |5 C||2 D| = 7**

**Dealer cards: |\* \*||8 D|**

**Do you want to 1)hit, 2)stand, or 3)split :** 1

**Drawing a card:**

**You drew: |5 D|**

**Your hand is: |5 D||5 C||2 D| = 12**

**Do you want to 1)hit, 2)stand, or 3)split :** 1

**Drawing a card:**

**You drew: |A S|**

**Your hand is: |A S||5 D||5 C||2 D| = 13**

**Do you want to 1)hit, 2)stand, or 3)split :** 1

**Drawing a card:**

**You drew: |6 S|**

**Your hand is: |6 S||A S||5 D||5 C||2 D| = 19**

**Do you want to 1)hit, 2)stand, or 3)split :** 2

**You have picked to stand**

**Dealer cards: |A C||8 D| = 19**

**Player total :19. Dealer total: 19**

**You tied. You lost $100. Your new balance is 900**

**Would you like to play again Y/N? :** n

**Thank you for playing**

**Your balance is: $900. Go to the cashier to get your money.**

**Your total earning is: $-100**

**UML Diagram**

