***California State University, Long Beach***

**CECS 282** *C++ for Java Programmers*

Final Project: Blackjack Slot Machine

December 15, 2017

Steven Garcia & Gabriel Flores

Introduction

The purpose of this program is to simulate a Blackjack Slot Machine. The user is prompted for an account number and the amount of money to bet. Two user cards will then be selected randomly and will be shown to the user. Another two cards will be randomly selected for the dealer’s hand. The program will display the value of the user’s cards and will prompt one of the following actions: hit, stand, or split. If the value of the user’s cards is less than or equal to 21 and worth more than the dealer’s, the user receives double the betted amount. If the values of the cards are the same, there is a tie and the money is split. Otherwise, the user wins nothing. The user can choose to continue playing after each hand.

Program Analysis

Our final project consists of Player, Account, Hand, and Card classes. The Player class represents a player in the blackjack game, the Account class represents the Player’s account, and the Hand class encapsulates the Card class to represent the Player’s blackjack hand.

Player class has get and setter methods for accessing attributes for their corresponding Account and Hand objects. The Player class also has a bet() method to bet a certain amount, a stand() method to stand, a hit() method to hit and get another card, and a split() method to split his or her hand. The usingSplitHand variable is a Boolean that indicates whether to add a card to the primary hand or the secondary hand. The player also has methods winGame(), loseGame(), and tieGame() to indicate whether or not he or she wins.

The Account class has get and setter methods for its attributes. It has private variables accountNumber and totalAmount to store the account number and amount in the account.

The Hand class has private variables cards, valueOfCards, numberOfCards, and cardMemoryCapacity. It stores the Card objects by having the cards variable point to a dynamic Card array. It has the appropriate get and setter methods to retrieve and manipulate the value and number of cards. It also has the allocateDynamicMemory() function to handle a possible memory leak.

The Card class has the get and setter methods to access the cardType and value. The value of the Card is determined by the cardType and is based off of the traditional set of 13 cards.

The program starts by asking the user for his or her account number and the amount of money to bet. Afterwards, a Player object is created for both the user and the dealer. Then the program executes a while loop that begins by dealing two cards to both the user and the dealer. The cards are randomly generated by the generateRandomCard() function. Both the user’s and dealer’s total value from the cards are output to the screen so that the user is aware of the current cards. After dealing the initial cards, another while loop is executed, which prompts the user to enter a move using the getUserDecision() function. The dealer is set to always hit until the value of his or her cards is above 15. If either the user’s or the dealer’s cards go over 21, the other player wins and the user is prompted to play again with the playAgain() function. If the user decides to play again, the loop reiterates. Otherwise, the loop breaks and the program ends.

Program Code

//

// main.cpp

// BlackJackSlotMachine

//

// Created by Steven Garcia on 11/20/17.

// Copyright Â© 2017 Steven Garcia. All rights reserved.

//

#include <iostream>

#include <stdlib.h> // for rand() and srand() function

#include <exception>

#include <algorithm> // for transform()

#include <iomanip>

#include "Player.h"

#include "Account.h"

#include "Card.h"

using namespace std;

Card generateRandomCard();

const int POSSIBLE\_CARDS = 13;

bool playAgain();

string getUserDecision();

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

cout << fixed << showpoint << setprecision(2);

bool isInvalidAccount = true;

bool isInvalidAmount = true;

bool userWantsToPlay = true;

int decimalOccurances = 0;

string accountNumber;

double amountBeingGambled;

string userInputDecision;

Account userAccount;

Player dealer;

Player user;

/\* ----------------

Accept a valid user account number.

---------------- \*/

do {

cout << "Enter your account number: ";

try {

cin >> accountNumber;

for (int i = 0; i < accountNumber.length(); i++) {

if (48 > accountNumber[i] || 57 < accountNumber[i]) {

throw "Invalid Character";

}

}

isInvalidAccount = false;

userAccount.setAccountNumber(accountNumber);

} catch (char const\* exc) {

cout << "Invalid account number." << endl;

}

} while (isInvalidAccount);

/\* ----------------

Accept valid double precision amount that user wants to gamble.

---------------- \*/

cin.ignore();

do {

try {

cout << "Enter the amount of money you want to bet (Ex: 1150.49): ";

getline(cin, userInputDecision); // take in input as a string

for (int i = 0; i < userInputDecision.length(); i++) {

if (userInputDecision[i] == 46) {

decimalOccurances++;

if (decimalOccurances > 1) {

throw "Invalid Character, too many decimal points";

}

continue;

}

if (48 > userInputDecision[i] || 57 < userInputDecision[i]) {

throw "Invalid Character";

}

}

// str needs to be converted to c\_str then converted to double value

amountBeingGambled = atof(userInputDecision.c\_str());

userAccount.setTotalAmount( amountBeingGambled );

cout << endl; // Output formatting

cout << "The amount being gambled is: $" << amountBeingGambled << endl;

isInvalidAmount = false;

} catch (char const\* exc) {

cout << "Invalid amount entered. Please enter in format example: 2500.00 " << endl;

}

} while( isInvalidAmount );

user.setAccount(userAccount);

/\* ----------------

Game functionality.

---------------- \*/

cout << endl; // Output formatting

cout << "Welcome to Blackjack! \n" << endl;

// Loop Until User decides not to play

while (userWantsToPlay) {

user.bet(amountBeingGambled);

user.resetHand();

dealer.resetHand();

// generate initial four random cards, 2 for dealer, and 2 for player

for (int i = 0; i < 2; i++) {

Card newUserCard = generateRandomCard();

user.addCard( newUserCard );

Card newDealerCard = generateRandomCard();

dealer.addCard( newDealerCard );

}

// Infinite Loop will Break when Dealer or User loses/wins

while(true){

while(dealer.getValueOfCards() < 15 ){

// Dealer will continue to hit as long as cards are less than 15

Card newDealerCard = generateRandomCard();

dealer.hit(newDealerCard);

if(dealer.getValueOfCards() > 21){

break;

}

}

cout << "The value of the dealer's cards is " << dealer.getValueOfCards() << endl;

cout << "The value of your current hand is " << user.getValueOfCards() << endl;

// add input validation

userInputDecision = getUserDecision();

if(userInputDecision == "HIT"){

cout << endl; // Output formatting

cout << "Dealing card..." << endl; // Output formatting

Card newUserCard = generateRandomCard();

user.hit( newUserCard );

if(user.getValueOfCards() > 21){

cout << "The value of your hand is " << user.getValueOfCards() << endl;

user.loseGame();

break;

}

else if(user.getValueOfCards() == 21){

cout << "The value of your hand is " << user.getValueOfCards() << endl;

user.winGame();

break;

}

}

else if(userInputDecision == "SPLIT"){

// Need to add Player implementation of split scenario

Card card1 = generateRandomCard();

Card card2 = generateRandomCard();

user.split(card1, card2);

if(user.getValueOfCards() > 21){

user.loseGame();

break;

}

}

else if(userInputDecision == "STAND"){

user.stand(); // Arbitrary function; does nothing

if(dealer.getValueOfCards() > 21){

user.winGame();

break;

}

else{

if(user.getValueOfCards() < dealer.getValueOfCards()){

user.loseGame();

break;

}

else{

user.winGame();

break;

}

}

}

}

// Prompt user to play again

userWantsToPlay = playAgain();

}

return 0;

}

Card generateRandomCard() {

srand(time(0));

int randomCardIdentifier = 0;

// Loop ensures that 0 is not returned

while(randomCardIdentifier <= 0){

randomCardIdentifier = rand() % POSSIBLE\_CARDS;

}

//Card newCard( randomCardIdentifier );

Card\* newCard = new Card( randomCardIdentifier );

return \*newCard;

}

bool playAgain(){

char userChoice;

while(true){

// loop only breaks when given valid answer (Y/N)

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

cin >> userChoice;

if(userChoice == 'Y') return true;

else if (userChoice == 'N') return false;

else cout << "Please enter a valid answer (Y/N)" << endl;

}

}

string getUserDecision(){

string input;

while(true){

cout << "Do you want to HIT, STAND, or SPLIT? : ";

cin >> input;

transform(input.begin(), input.end(), input.begin(), ::toupper);

if(input == "HIT")

return "HIT";

else if(input == "SPLIT")

return "SPLIT";

else if(input == "STAND")

return "STAND";

else

cout << "Please enter a valid answer. " << endl;

}

}

//

// Player.hpp

// BlackJackSlotMachine

//

// Created by Steven Garcia on 11/20/17.

// Copyright Â© 2017 Steven Garcia. All rights reserved.

//

#ifndef Player\_h

#define Player\_h

#include "Account.h"

#include "Hand.h"

#include "Card.h"

#include <string>

class Player {

private:

Account playerAccount;

Hand hand;

Hand splitHand;

double betAmount;

bool hasSplit;

bool usingSplitHand;

public:

Player();

Player(Account);

~Player();

void setAccount(Account);

double getAccount() const;

double bet(double);

void resetHand();

bool isSplit() const;

void switchHands();

string getCurrentHand();

void setNumberOfCardsInHand(int);

int getNumberOfCardsInHand() const;

void addCard(Card&);

int getValueOfCards();

void stand();

void hit(Card&);

void split(Card&, Card&);

void winGame();

void loseGame();

void tieGame();

};

#endif /\* Player\_h \*/

//

// Account.hpp

// BlackJackSlotMachine

//

// Created by Steven Garcia on 11/20/17.

// Copyright Â© 2017 Steven Garcia. All rights reserved.

//

#ifndef Account\_h

#define Account\_h

#include <string>

using namespace std;

class Account {

private:

string accountNumber;

double totalAmount;

public:

Account();

Account(string);

~Account();

string getAccountNumber() const;

void setAccountNumber(string);

double getTotalAmount() const;

void setTotalAmount(double);

};

#endif /\* Account\_h \*/

#ifndef Hand\_h

#define Hand\_h

#include <string>

#include "Card.h"

using namespace std;

class Hand{

private:

Card\* cards = nullptr;

string type;

int valueOfCards;

int numberOfCards;

int cardMemoryCapacity;

public:

Hand();

Hand(string);

~Hand();

void addCard(Card&);

string getType() const;

void setType(string);

int getValueOfCards() const;

void setValueOfCards(int);

int getNumberOfCards() const;

void setNumberOfCards(int);

void allocateDynamicMemory();

};

#endif

//

// Card.hpp

// BlackJackSlotMachine

//

// Created by Steven Garcia on 11/20/17.

// Copyright Â© 2017 Steven Garcia. All rights reserved.

//

#ifndef Card\_h

#define Card\_h

#include <string>

using namespace std;

class Card {

private:

string cardType;

int value;

public:

Card();

Card(int);

void setCardType(int);

string getCardType() const;

int getValue() const;

void setValue(int);

};

#endif /\* Card\_h \*/

//

// Player.cpp

// BlackJackSlotMachine

//

// Created by Steven Garcia on 11/20/17.

// Copyright Â© 2017 Steven Garcia. All rights reserved.

//

#include "Player.h"

#include "Card.h"

#include <iostream>

using namespace std;

Player::Player() {

this->hasSplit = false;

this->usingSplitHand = false;

this->betAmount = 0;

}

Player::Player(Account accountNumber) {

this->playerAccount = accountNumber;

this->hasSplit = false;

this->usingSplitHand = false;

this->betAmount = 0;

}

Player::~Player(){

}

void Player::setAccount(Account userAccount) {

this->playerAccount = userAccount;

}

double Player::getAccount() const {

return playerAccount.getTotalAmount();

}

double Player::bet(double amount){

this->playerAccount.setTotalAmount(this->playerAccount.getTotalAmount() - amount);

this->betAmount += amount;

return this->betAmount;

}

void Player::resetHand(){

// !!!Potential Memory Leak!!!

Hand\* newHand = new Hand();

this->hand = \*newHand;

}

void Player::switchHands(){

usingSplitHand = !usingSplitHand;

}

bool Player::isSplit() const{

return hasSplit;

}

int Player::getNumberOfCardsInHand() const {

if(usingSplitHand){

return splitHand.getValueOfCards();

}

else{

return hand.getValueOfCards();

}

}

string Player::getCurrentHand(){

if(usingSplitHand){

return splitHand.getType();

}

else{

return hand.getType();

}

}

void Player::addCard(Card& newCard) {

hand.addCard(newCard);

}

int Player::getValueOfCards() {

if(usingSplitHand){

return splitHand.getValueOfCards();

}

else{

return hand.getValueOfCards();

}

}

void Player::stand() {

// Player does nothing

}

void Player::hit(Card& newCard) {

if(usingSplitHand) {

splitHand.addCard(newCard);

}

else{

hand.addCard(newCard);

}

}

void Player::split(Card& handCard, Card& splitCard) {

this->hasSplit = true;

bet(this->betAmount); // Double the bet

hand.addCard(handCard);

splitHand.addCard(splitCard);

splitHand.setType("Secondary");

}

void Player::winGame() {

this->playerAccount.setTotalAmount(this->playerAccount.getTotalAmount() + (2 \* this->betAmount));

cout << "You win $" << betAmount << endl;

betAmount = 0;

}

void Player::loseGame() {

cout << "You lose $" << betAmount << endl;

betAmount = 0;

}

void Player::tieGame(){

// User bet is split in half

this->playerAccount.setTotalAmount(this->playerAccount.getTotalAmount() + (this->betAmount / 2));

betAmount = 0;

}

//

// Account.cpp

// BlackJackSlotMachine

//

// Created by Steven Garcia on 11/20/17.

// Copyright Â© 2017 Steven Garcia. All rights reserved.

//

#include "Account.h"

Account::Account() {

accountNumber = "";

totalAmount = 0.0;

}

Account::Account(string accountNumber) {

this->accountNumber = accountNumber;

}

Account::~Account(){

}

string Account::getAccountNumber() const {

return this->accountNumber;

}

void Account::setAccountNumber(string accountNumber) {

this->accountNumber = accountNumber;

}

//

// Card.cpp

// BlackJackSlotMachine

//

// Created by Steven Garcia on 11/20/17.

// Copyright Â© 2017 Steven Garcia. All rights reserved.

//

#include "Card.h"

using namespace std;

Card::Card() {

}

/\*\*

This constructor allows for instantiating

of any one of possible cards based on the

argument of a randomly generated number.

\*\*/

Card::Card(int randomNum) {

setCardType(randomNum);

}

string Card::getCardType() const {

return this->cardType;

}

void Card::setCardType(int randomNum) {

switch (randomNum) {

case 1:

// ace

this->cardType = "Ace";

this->value = 1; // 1 or 11 based on hand

break;

case 2:

this->cardType = "Face Value";

this->value = 2;

break;

case 3:

this->cardType = "Face Value";

this->value = 3;

break;

case 4:

this->cardType = "Face Value";

this->value = 4;

break;

case 5:

this->cardType = "Face Value";

this->value = 5;

break;

case 6:

this->cardType = "Face Value";

this->value = 6;

break;

case 7:

this->cardType = "Face Value";

this->value = 7;

break;

case 8:

this->cardType = "Face Value";

this->value = 8;

break;

case 9:

this->cardType = "Face Value";

this->value = 9;

break;

case 10:

this->cardType = "Face Value";

this->value = 10;

break;

case 11:

this->cardType = "King (FaceCard)";

this->value = 10;

break;

case 12:

this->cardType = "Queen (FaceCard)";

this->value = 10;

break;

case 13:

this->cardType = "Jack (FaceCard)";

this->value = 10;

break;

}

}

int Card::getValue() const {

return this->value;

}

void Card::setValue(int value) {

this->value = value;

}

double Account::getTotalAmount() const {

return this->totalAmount;

}

void Account::setTotalAmount(double totalAmount) {

this->totalAmount = totalAmount;

}

#include "Hand.h"

#include <iostream>

using namespace std;

Hand::Hand(){

this->type = "Primary";

this->numberOfCards = 0;

this->valueOfCards = 0;

this->cardMemoryCapacity = 10;

this->cards = new Card[cardMemoryCapacity];

}

Hand::Hand(string handType){

this->type = handType;

this->numberOfCards = 0;

this->valueOfCards = 0;

this->cardMemoryCapacity = 10;

this->cards = new Card[cardMemoryCapacity];

}

Hand::~Hand(){

delete cards;

}

void Hand::addCard(Card& newCard){

// ensure have enough memory in dynamic array

if(this->numberOfCards < this->cardMemoryCapacity){

\*(cards + numberOfCards + 1) = newCard;

if(newCard.getCardType() == "Ace"){

// Checks the value that a new Ace should be: 1 (default), or 11

if(this->valueOfCards + 11 == 21)

setValueOfCards(this->valueOfCards + 11);

}

else{

setValueOfCards(this->valueOfCards + newCard.getValue());

}

this->numberOfCards++;

}

else{

// allocate internal memory

allocateDynamicMemory();

cout << "Card Memory Capacity exceeded. Unable to add new card. " << endl;

}

}

string Hand::getType()const{

return type;

}

void Hand::setType(string handType){

type = handType;

}

int Hand::getValueOfCards() const{

return this->valueOfCards;

}

void Hand::setValueOfCards(int value){

this->valueOfCards = value;

}

int Hand::getNumberOfCards() const{

return this->numberOfCards;

}

void Hand::setNumberOfCards(int num){

this->numberOfCards = num;

}

void Hand::allocateDynamicMemory() {

cardMemoryCapacity \*= 2;

Card\* newCardPtr = new Card[cardMemoryCapacity];

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

\*(newCardPtr + i) = cards[i];

}

delete cards;

cards = newCardPtr;

}

Sample Output

Enter your account number: 2000

Enter the amount of money you want to bet (Ex: 1150.49): 100

The amount being gambled is: $100.00

Welcome to Blackjack!

The value of the dealer's cards is 16

The value of your current hand is 16

Do you want to HIT, STAND, or SPLIT? : HIT

Dealing card...

The value of your hand is 21

You win $100.00

Would you like to play again (Y/N)?

Y

The value of the dealer's cards is 15

The value of your current hand is 10

Do you want to HIT, STAND, or SPLIT? : STAND

You lose $100.00

Would you like to play again (Y/N)?

N

------------------------------

Enter your account number: 2000

Enter the amount of money you want to bet (Ex: 1150.49): 100

The amount being gambled is: $100.00

Welcome to Blackjack!

The value of the dealer's cards is 16

The value of your current hand is 4

Do you want to HIT, STAND, or SPLIT? : HIT

Dealing card...

The value of the dealer's cards is 16

The value of your current hand is 6

Do you want to HIT, STAND, or SPLIT? : HIT

Dealing card...

The value of the dealer's cards is 16

The value of your current hand is 15

Do you want to HIT, STAND, or SPLIT? : SPLIT

The value of the dealer's cards is 16

The value of your current hand is 21

Do you want to HIT, STAND, or SPLIT? : STAND

You win $200.00

Would you like to play again (Y/N)?

Y

The value of the dealer's cards is 18

The value of your current hand is 18

Do you want to HIT, STAND, or SPLIT? : aa

Please enter a valid answer.

Do you want to HIT, STAND, or SPLIT? : aa

Please enter a valid answer.

Do you want to HIT, STAND, or SPLIT? : SPLIT

The value of the dealer's cards is 18

The value of your current hand is 21

Do you want to HIT, STAND, or SPLIT? : STAND

You win $200.00

Would you like to play again (Y/N)?

N

--------------------------------

Enter your account number: 100

Enter the amount of money you want to bet (Ex: 1150.49): 200

The amount being gambled is: $200.00

Welcome to Blackjack!

The value of the dealer's cards is 15

The value of your current hand is 10

Do you want to HIT, STAND, or SPLIT? : HIT

Dealing card...

The value of the dealer's cards is 15

The value of your current hand is 18

Do you want to HIT, STAND, or SPLIT? : HIT

Dealing card...

The value of the dealer's cards is 15

The value of your current hand is 20

Do you want to HIT, STAND, or SPLIT? : HIT

Dealing card...

The value of your hand is 30

You lose $200.00

Would you like to play again (Y/N)?

N

--------------------------------