Project 2: Blackjack

By: Hamza Khan

Class: CSC 17A

Date Due: December 12, 2014

Professor Lehr

Table of Contents

1. Introduction…………………………………………………………………………....3
2. Summary……………………………………………………………………………...3
3. Description…………………………………………………………………………...3
4. Sample Input and Output…………………………………………………................3
5. Flowchart……………………………………………………………………………..4
6. Variables……………………………………………………………………………...5
7. Reference……………………………………………………………………………..5
8. Code…………………………………………………………………………………..5-18
9. Introduction

For this project I created a game of Blackjack from scratch because my original RPG game kept crashing once I had opened up the original source code for the game. Also since I had to create a game for my CSC 11 class I decided to make both projects hand in hand and edited the CSC 11 game a bit.

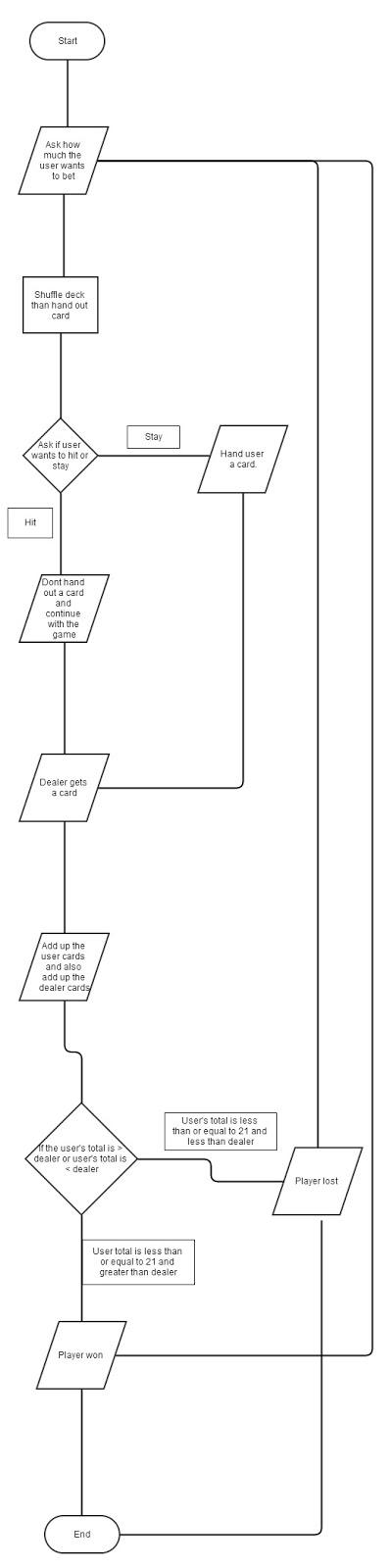
1. Summary

|  |  |
| --- | --- |
| Number of lines in main | 332 |
| Number of blank spaces in main | 11 |
| Number of comment lines in main | 146 |
| Total in main | 489 |

1. Description
2. Sample input and output

Unfortunately I can’t include any sample screenshots off the game because the snipping tool on my PC is running into issues.

1. Flowchart



1. Variables
2. Classes

I used classes to store the different card values in there. Then the class also shuffles the deck and store it and distribute the cards. I used classes in files card.h, deck.h, hand.h, and player.h.

1. Templates

I used templates to allow the game to set itself up. I used classes in files player.h and table.h

1. Strings

I used strings to store a users name so that if they wanted to play the game again all they have to do is type in their name and it’ll pull your game data if you played the game before. I used strings in lines 65 - 75 in the main.cpp file.

1. Reference

All the code in this game is originally my own creation. The only outside source I used was the class textbook to create this game.

1. Code (main)

#include <cstdlib>

#include <iostream>

#include <ctime>

using namespace std;

#include "Player.h"

#include "Deck.h"

#include "Hand.h"

//Enums

/\*\* \brief Makes the passing of end game details easier to understand \*/

enum

{

BJWIN = 0, WIN = 1, LOSE = 2, PUSH = 2

};

//Function Prototypes

string intro();

void blackJack(Player&);

void deal(Player&, Hand&, Deck&);

void bet(Player&);

void table(Hand,Hand,bool);

void table(Hand,Hand,Hand,bool);

bool insurance(Player&,Hand);

bool choices(Player&, Deck&);

bool hitOrStay(Player&,Deck&, bool);

bool dealerPlay(Hand&, Deck&);

void resolve(Player&, Hand);

int getInt(string);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\*\*

\* Used to seed random, initialize the player and then start the game

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(int argc, char\*\* argv) {

//Seed Random

srand(time(0));

/\*\* \brief Stores the name of the sessions current player \*/

string name = intro();

/\*\* \brief Creates the player object by passing 'name' into the constructor \*/

Player p(name);

//Print player data

p.printData();

//Player Blackjack

blackJack(p);

//Stay Execution

return 0;

}

/\*\*

\* \brief Displays greeting and prompts user to enter name

\* @return Returns the player's name as input by the user

\*/

string intro(){

cout<<"||====================================================||"<<endl

<<"|| Blackjack ||"<<endl

<<"||====================================================||"<<endl;

cout<<"|| Please Enter your name ||"<<endl

<<"|| (returning players use previous name) ||"<<endl

<<"||====================================================||"<<endl;

cout<<"Name: ";

/\*\* \brief Stores the user input for name \*/

string name;

getline(cin,name);

return name;

}

/\*\*

\* \brief Plays the game of blackjack

\* @param p Player from main by reference

\*/

void blackJack(Player &p){

//Declare

/\*\* \brief Flags true if player is done with the turn \*/

bool endHand = false;

/\*\* \brief Flags true if initial hand revealed Blackjack for player or dealer

\* Used to prevent the dealer from playing later in the function \*/

bool bJack;

/\*\* \brief BLACKJACK happens when value of hand is 21

\* used to control Blackjack and Bust situations \*/

const int BLACKJACK = 21;

/\*\* \brief Stores value for the max cards any hand can have \*/

const int MAXHAND = 6;

do{

/////////////////////////////////////////////////////////

///// STEP 0: SETUP THE GAME /////////////////////////

/////////////////////////////////////////////////////////

/\*\* \brief Instantiates Deck with Cards and shuffles \*/

Deck deck;

//Deal two cards

p.deleteHand();

/\*\* \brief Stores the dealers hand \*/

Hand dealer;

deal(p,dealer,deck);

//Get the bet and put it on the table

endHand = false;

bJack = false;

bet(p);

//Show cards and options

table(p.getHand(),dealer,true);

if(p.getBet() == 0){

endHand = true;

}

/////////////////////////////////////////////////////////

///// STEP 1: CHECK FOR PLAYER BLACKJACK /////////////

/////////////////////////////////////////////////////////

//Player has blackjack

if(!endHand&&p.getHandValue()==BLACKJACK){

cout<<"PLAYER SHOWS BLACKJACK!!!"<<endl;

endHand = true;

bJack = true;

//and dealer doesn't

if(dealer.getValue()!=BLACKJACK){

cout<<"PLAYER WINS!!!"<<endl;

p.end(BJWIN);

}

//and dealer does

else if(dealer.getValue()==BLACKJACK){

cout<<"but so does DEALER...PLAYER pushes"<<endl;

p.end(PUSH);

}

}

/////////////////////////////////////////////////////////

///// STEP 2: CHECK FOR DEALER BLACKJACK /////////////

/////////////////////////////////////////////////////////

//Dealer Shows Ace

if(!endHand&&dealer.aceUp())

endHand = insurance(p,dealer);

/////////////////////////////////////////////////////////

///// STEP 3: SURRENDER, SPLIT OR DOUBLEDOWN /////////

/////////////////////////////////////////////////////////

//The player may surrender and forfeit half of their bet

if(!endHand){

endHand = choices(p,deck);

table(p.getHand(),dealer,true);

}

/////////////////////////////////////////////////////////

///// STEP 4: PLAYER HAND ////////////////////////////

/////////////////////////////////////////////////////////

//Play First Hand

while(!endHand){

cout<<"||====================================================||"<<endl

<<"|| MAIN HAND PLAY ||"<<endl;

endHand = hitOrStay(p,deck,false);

if(p.getHandSize()==6) //More than six cards

endHand = true;

if(p.doubleD&&p.getHandSize()==3) //More than 3 cards if doubled down

endHand = true;

if(p.getHandValue()>BLACKJACK) //Busted

endHand = true;

table(p.getHand(),dealer,true);

cout<<endl<<endl;

}

if(p.didSplit){

cout<<"||====================================================||"<<endl

<<"|| SPLIT HAND PLAY ||"<<endl;

bool splitEnd = false;

while(!splitEnd&&p.getSplitSize()!=6){

splitEnd = hitOrStay(p,deck,true);

if(p.getSplitValue()>BLACKJACK)

splitEnd = true;

table(p.getHand(),p.getSplit(),dealer,true);

}

}

/////////////////////////////////////////////////////////

///// STEP 5: DEALER PLAY ////////////////////////////

/////////////////////////////////////////////////////////

if(!bJack)

if(!(p.getHandValue()>BLACKJACK)||(p.didSplit&&!(p.getSplitValue()>BLACKJACK))){

while(dealerPlay(dealer,deck));

table(p.getHand(),dealer,false);

}

/////////////////////////////////////////////////////////

///// STEP 6: RESOLVE GAME ///////////////////////////

/////////////////////////////////////////////////////////

resolve(p,dealer);

/////////////////////////////////////////////////////////

///// STEP 7: PRINT STATS ////////////////////////////

/////////////////////////////////////////////////////////

p.anotherHand();

p.printData();

}while(p.getBet()!=0);

}

/\*\*

\* \brief Deals alternating cards to the player and dealer until both have two

\* @param p Player by reference

\* @param dealer Hand by reference

\* @param deck Deck by reference

\*/

void deal(Player &p, Hand &dealer, Deck &deck){

p.hit(deck.deal());

dealer.getCard(deck.deal());

p.hit(deck.deal());

dealer.getCard(deck.deal());

}

/\*\*

\* \brief Prompts the Player for their bet and passes the result to the player object for handling

\* @param p Player by reference

\*/

void bet(Player &p){

/\*\* stores the bet input by the player\*/

float bet;

cout<<"|| Enter your bet: ";

cin>>bet;

p.setBet(bet);

}

/\*\*

\* \brief Prints the Player and Dealer's hand

\* @param p Hand

\* @param dealer Hand

\* @param hide True if the dealer has a card face down

\*/

void table(Hand player,Hand dealer,bool hide){

if(!hide){

cout<<"||====================================================||"<<endl;

cout<<"|| Player Hand: "<<endl<<"|| ";

player.printValue();

cout<<"|| Dealer Hand: "<<endl<<"|| ";

dealer.printValue();

cout<<"||====================================================||"<<endl;

}

else{

cout<<"||====================================================||"<<endl;

cout<<"|| Player Hand: "<<endl<<"|| ";

player.printValue();

cout<<"|| Dealer Hand: "<<endl<<"|| ";

dealer.printOne();

cout<<"||====================================================||"<<endl;

}

}

/\*\*

\* \brief Same functionality as table except to also display a player's split hand

\* @param p Hand

\* @param s Split Hand

\* @param dealer Hand

\* @param hide True if the dealer has a card face down

\*/

void table(Hand p, Hand s, Hand dealer, bool hide){

if(!hide){

cout<<"||====================================================||"<<endl;

cout<<"|| Player Hand: "<<endl<<"|| ";

p.printValue();

cout<<"|| Split Hand: "<<endl<<"|| ";

s.printValue();

cout<<"|| Dealer Hand: "<<endl<<"|| ";

dealer.printValue();

cout<<"||====================================================||"<<endl;

}

else{

cout<<"||====================================================||"<<endl;

cout<<"|| Player Hand: "<<endl<<"|| ";

p.printValue();

cout<<"|| Split Hand: "<<endl<<"|| ";

s.printValue();

cout<<"|| Dealer Hand: "<<endl<<"|| ";

dealer.printOne();

cout<<"||====================================================||"<<endl;

}

}

/\*\*

\* \brief Displays 1st turn choices

\* This separate function for the first turn allows a player to choose moves

\* specific to the first turn such as double down, split and surrender

\* @param p Player

\* @param d Deck

\* @return returns true if the game is ended by choices in this function

\*/

bool choices(Player &p, Deck &d){

/\*\* used to save player's choice from the menu\*/

int choice;

/\*\* return from function to signify the game is over \*/

bool gameover = false;

do{

cout<<"||====================================================||"<<endl

<<"|| [1] Hit ||"<<endl

<<"|| [2] Stay ||"<<endl

<<"|| [3] Double Down ||"<<endl

<<"|| [4] Split ||"<<endl

<<"|| [0] Surrender Game ||"<<endl

<<"||====================================================||"<<endl;

choice = getInt("Choice: ");

if(choice==4&&!p.canSplit()){

cout<<"Splitting is only possible if you have two cards of the same value."<<endl;

choice = 5;

}

}while(choice!=2&&choice!=1&&choice!=0&&choice!=3&&choice!=4);

if(choice==0){

p.surrender();

cout<<"PLAYER has surrendered the round and half of their bet..."<<endl;

gameover = true;

}

else if(choice==1){

p.hit(d.deal());

}

else if(choice==2){

gameover = true;

}

else if(choice==3){

p.doubleDown();

cout<<"PLAYER has doubled their bet for an additional card"<<endl;

p.hit(d.deal());

gameover = true;

}

else if(choice==4){

p.split();

cout<<"PLAYER has split their hand into two hands"<<endl;

}

return gameover;

}

/\*\*

\* \brief Prompts user to buy insurance if the dealer shows an Ace

\* @param p Player

\* @param d Deck

\* @return returns true if the dealer actually has Blackjack

\*/

bool insurance(Player &p, Hand dealer){

/\*\* Stores players choice to buy or not buy insurance\*/

int choice;

/\*\* Is true if the round is over\*/

bool gameover = false;

do{

cout<<"||====================================================||"<<endl

<<"|| [1] Buy Insurance ||"<<endl

<<"|| [2] Don't Buy Insurance ||"<<endl

<<"||====================================================||"<<endl;

choice = getInt("Choice: ");

}while(choice!=2&&choice!=1);

//If player buys insurance (dealer no blackjack)

if(choice==1&&dealer.getValue()!=21){

cout<<"|| Player bought insurance, but it didn't payoff..."<<endl;

p.insurance();

}

//If player buys insurance (dealer blackjack)

else if(choice==1&&dealer.getValue()==21){

cout<<"|| Player bought insurance, AND IT PAID OFF!!!"<<endl;

gameover = true;

}

//If player doesn't buy insurance (dealer no blackjack)

else if(choice==2&&dealer.getValue()!=21){

cout<<"|| Player denied insurance, dealer did not show Blackjack."<<endl;

}

//If player doesn't buy insurance (dealer blackjack)

if(choice==2&&dealer.getValue()==21){

cout<<"|| Player denied insurance, dealer shows Blackjack..."<<endl;

gameover = true;

}

return gameover;

}

/\*\*

\* \brief Offers the choice to hit or stay on the players hand

\* @param p Player

\* @param dealer Dealer's Hand

\* @return returns true if the player stays on his hand

\*/

bool hitOrStay(Player &p,Deck &d, bool split){

/\*\* Stores players choice to buy or not buy insurance\*/

int choice;

/\*\* Is true if the round is over\*/

bool gameover = false;

if(!split){

do{

cout<<"||====================================================||"<<endl

<<"|| [1] Hit ||"<<endl

<<"|| [2] Stay ||"<<endl

<<"||====================================================||"<<endl;

choice = getInt("Choice: ");

}while(choice!=1&&choice!=2);

if(choice==1){

p.hit(d.deal());

}

else if(choice==2){

gameover = true;

}

}

else if(split){

do{

cout<<"||====================================================||"<<endl

<<"|| [1] Hit ||"<<endl

<<"|| [2] Stay ||"<<endl

<<"||====================================================||"<<endl;

choice = getInt("Choice: ");

}while(choice!=1&&choice!=2);

if(choice==1){

p.hitSplit(d.deal());

}

else if(choice==2){

gameover = true;

}

}

return gameover;

}

/\*\*

\* \brief Resolves the game by dishing out money and cleaning up variables

\* @param p Player

\* @param d Deck

\* @param split True if hitting or staying on the split hand

\* @return gameover

\*/

void resolve(Player &p, Hand dealer){

const int BLACKJACK = 21;

cout<<"||====================================================||"<<endl;

if(p.getHandValue()>BLACKJACK){

p.end(LOSE);

cout<<"|| PLAYER BUSTS!!! ||"<<endl;

}

else if(dealer.getValue()>BLACKJACK){

p.end(WIN);

cout<<"|| DEALER BUSTS, PLAYER WINS!!! ||"<<endl;

}

else if(p.getHandValue()==dealer.getValue()){

p.end(PUSH);

cout<<"|| PLAYER PUSHES... ||"<<endl;

}

else if(p.getHandValue() > dealer.getValue()){

p.end(WIN);

cout<<"|| PLAYER WINS!!! ||"<<endl;

}

else if(p.getHandValue() < dealer.getValue()){

p.end(LOSE);

cout<<"|| PLAYER LOST... ||"<<endl;

}

}

/\*\*

\* \brief Plays for the dealer

\* @param dealer Hand

\* @param d Deck

\* @return Returns true if the dealer is done playing

\*/

bool dealerPlay(Hand &dealer, Deck &d){

/\*\* Dealer won't hit on a hand higher than 17 \*/

const int MAXDEALER = 17;

cout<<"||====================================================||"<<endl

<<"|| Dealer's Turn ||"<<endl

<<"||====================================================||"<<endl;

if(dealer.getValue()<MAXDEALER){

dealer.getCard(d.deal());

return true;

}

else

return false;

}

/\*\*

\* \brief Returns an int based on a string

\* @param dealer Hand

\* @param d Deck

\* @return Returns an int based on the passed string

\*/

int getInt(string s){

//Declare

/\*\* Stores the integer to be returned \*/

int integer;

/\*\* Is set to true if the string doesn't resemble an int \*/

bool fail;

//Some magic to make sure an integer is entered

do{

cout<<s;

cin>>integer;

fail = false;

if(cin.fail()||cin.peek()=='.'){

fail = true;

cin.clear();

string dummy;

cin>>dummy;

cout<<"INVALID INPUT...PLEASE ENTER INTEGER."<<endl;

}

cin.ignore();

}while(fail);

return integer;

}