Project 2

<Blackjack>

CSC-17A-45398

Name: Juan Enriquez

Date: 7/25/2021

Introduction

Title: Blackjack

This is a simple card game of Blackjack which can have up to 3 players plus the dealer. So there are at least 2 or maximum of 4 players at a time.

The reason why I am doing this is because it is for my class project, and this is important for me to do because this project helps me use the concepts that we have gone over in class. For this version, it will be made of classes instead of structures.

Summary

Project size: about 1032 lines

Amount of variables: 11

Amount of objects: 5 (2 of them object arrays)

Amount of classes: 5

Amount of header files: 7

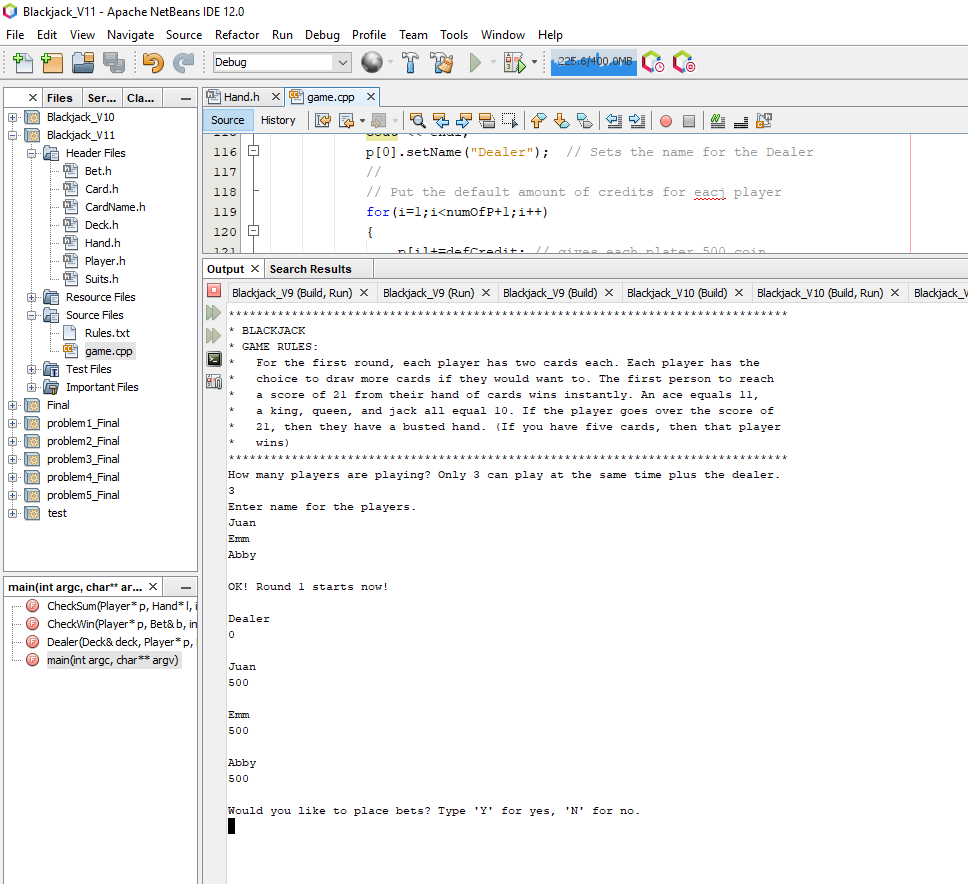
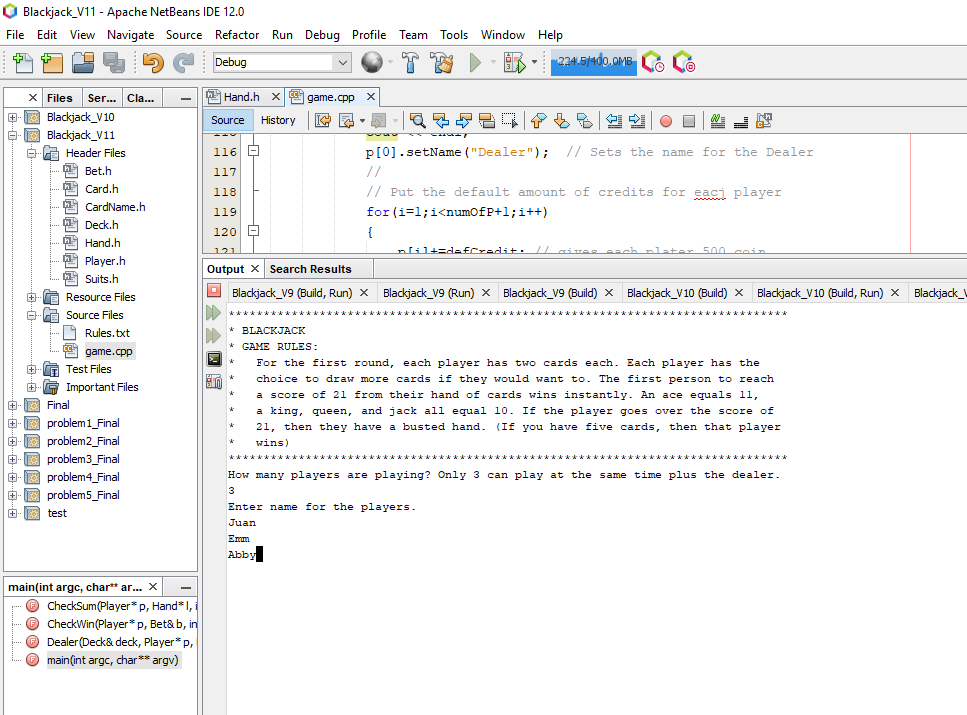
Amount of time taken: about 10 hours

This project was a challenge to do since it would be my first time using object oriented methods and implementing it to what I had previously, which were structures. What I had to do was study code to see the correct way to go about OOP. I still need to get accustomed to it but im sure it will not take long.

Some things that I found out is that I can make destructors to destroy objects and you can initialize the object again within a while statement. With this discovery, I figured that I can use this to delete the contents of the hand and the deck as well as the players info if the used wanted to completely restart the game.

Description

I programmed my project so that everything that the user types, it would be transferred into each respective class and applied to the correct functions. What is then done is outputting the correct output.



Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generatedText

Description automatically generated

Graphical user interface, text, application

Description automatically generated

At the end, I need to change the position of the total so that it can reset with the hand. If I don’t, then the wrong output will be read since the Player class is never deleted since it only deletes when user totally resets the game.

Flow Chart (Read from left to right)

Diagram

Description automatically generated

Diagram, schematic

Description automatically generated

Diagram, schematic

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generatedDiagram, schematic

Description automatically generated

Variables of note

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Var Name | Description | Location |
| Integer | numOfP | Tells how many people are playing | int main() |
| index | Gives the position of the deck | int main(), PrintAll(), SetupCards(), draw(), Dealer() |
| enum | Suit | Has the contents of the card type | Suit.h |
| CardName | Has the contents of the card values | CardName.h |
| class | Card | Holds the contents of each card | Card.h |
| Deck | Has an array of the Card structure that holds 52 objects | Deck.h |
| Player | Holds the contents of the info of the player | Player.h |
|  | Hand | Holds the contents of the hand of each player | Bet.h |
| srand | (time(0)) | Random generator | Int main()//line 86 |
| fstream | rules | Used to open the file Rules.txt | int main()//Line 94 |

**Concepts**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chapter** | **Section** | Checklist | Location |
| **13** |  | **Classes** |  |
|  | **1 to 3** | **Instances of a Class** | int main(), Line  **40 to 45** |
|  | **4** | **Private Data Members** | **Player.h, Bet.h, Deck.h** |
|  | **5** | **Specification vs. Implementaion** | **Player.h //line 66, Bet.h, Deck.h, Card.h** |
|  | **6** | **Inline** | **Bet.h, Deck.h** |
|  | **7,8,10** | **Constructors** | **Player.h, Bet.h, Deck.h, Card.h** |
|  | **9** | **Destructors** | **Bet.h, Deck.h** |
|  | **12** | **Array Of Objects** | **int main(), //Line 44 & 45** |
|  | **16** | **UML** | **Shown in the flowchart section. Also in the Project2Flowchart.pdf** |
| **14** |  | **More about Classes** |  |
|  | **1** | **Static** | **Deck.h //Line 22** |
|  | **2** | **Friends** | **Player.h, // Line 70** |
|  | **4** | **Copy Constructors** |  |
|  | **5** | **Operator Overloading** | **Player.h // line 70** |
|  | **7** | **Aggregation** | **Deck.h // line 21** |
| **15** |  | **Inheritence** |  |
|  | **1** | **Protected members** | **Player.h // Line 23** |
|  | **2 to 5** | **Base Class to Derived** | **Deck.h // Line 18** |
|  | **6** | **Polymorphic associations** | **Player.h // Line 36** |
|  | **7** | **Abstract Classes** |  |
| **16** |  | **Advanced Classes** |  |
|  | **1** | **Exceptions** |  |
|  | **2 to 4** | **Templates** |  |
|  | **5** | **STL** |  |

**Pseudo code**

**Classes**

**Create Cards Class:**

Create variables to hole the cards type and value,

Insert them into a class array

**Create Players Class:**

**Input Players-** Create a class that holds the contents of each player such as the total credits of each player.

**Create Bet Class:**

Ask for bets within the member functions inside of the Bet class which would is used only for the member

function PlaceBets().

**Create Deck Class:**

Create class which hold member functions that create the deck and let players draw from the deck

**Enter Main Program:**

Initialize variables to access structures,

Make a file that has the output for the rules,

Output file Rules.txt,

Ask user how many players they want,

while {

the amount is not correct, tell user that input is incorrect and to try again

},

Ask user for each players name,

**SetupCards** Function call that sets up the card deck,

Output the contents of each players hand,

**Dealer** Function gets called so that each player can draw cards, Dealer also draws,

**CheckSum** Function call gets called which checks the sum of each players hand,

**CheckWin** Function call used to check the win conditions of each player including Dealer

**Exit Program;**

**Create Functions for the program:**

**SetupCards(){**

Go through the contents of the structure array of cards and set the deck,

**Shuffle** function call to shuffle the deck

**}**

**PlaceBets(){**

This is called from the Bet class so that players can place bets.

**}**

**Dealer(){**

Deal cards to each player in a menu. The dealer also draws until sum of hand reaches

17 or over

**}**

**CheckSum(){**

Function checks the sum of each players hand

**{**

**CheckWin(){**

Check each players hand sum and compare scores with the dealer

**}**

**Shuffle(){**

Shuffles the array of cards in the Deck structure array

**}**

**PrintCard(){**

Prints the contents of the contents the **Deck** structure array

**}**

**PrintSuit(){**

Defines the type of card

**}**

**PrintValue(){**

Defines the value of card

**}**

**Draw(){**

Draws a card,

Index decrements the deck position,

**}**

**References**

1. **Textbook**
2. **Lecture**
3. **YouTube** 
   1. (280) [C++ Tutorial #11.2] Ace Combinations - GameDev Prep Course – YouTube
   2. (280) [C++ Tutorial #11.1] Instantiating a Deck of Cards – YouTube
4. Reddit

**Program**

**/\***

**\* File: game.cpp**

**\* Author: Juan Enriquez**

**\* Purpose: To create a program that plays the Blackjack card game**

**\* Version: 11**

**\* Created on July 13, 2021, 6:03 PM**

**\*/**

**//System Libraries - Post Here**

**#include <iostream>**

**#include <string>**

**#include <fstream>**

**#include <iomanip>**

**#include <cstdlib>**

**#include <ctime>**

**using namespace std;**

**// after done with project, redo and put structs/enums into seperate files**

**//User Libraries - Post Here**

**#include "Suits.h"**

**#include "CardName.h"**

**#include "Card.h"**

**#include "Deck.h"**

**#include "Player.h"**

**#include "Bet.h"**

**#include "Hand.h"**

**//Function Prototypes - Post Here**

**void Dealer(Deck &, Player \*,Hand \*,int, int &, int &, int &, int &);**

**void CheckSum(Player \*,Hand \*,int,int,int,int);**

**void CheckWin(Player \*,Bet &,int);**

**//Execution Begins Here**

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

**//**

**// random generator seed**

**srand(time(0)); // required to make the random operator work**

**//**

**// Object Variables**

**Card cards;**

**Bet put;**

**// Create a dynamic class for people**

**Deck deck; //makes object for the deck**

**Player \*p; //makes object for players**

**Hand \*load;**

**//**

**// Variables**

**fstream rules; // set up to open file for da rules**

**string fileName="Rules.txt"; // c-string thing**

**string readLine;**

**string answer=""; //used to ask user if they want to play**

**string restart="";**

**float defCredit=500.00; //Gives the default credit to each player**

**int numOfP; // Number of players**

**int i; //counter**

**int j; //counter**

**int round; //counter to distingish rounds**

**int index=52; // index of the deck**

**//**

**// Open file to print out rules**

**rules.open(fileName, ios::in | ios::out); // opens a file and put it into input mode**

**while(getline(rules, readLine)) // while loop used to print out contents**

**{**

**cout << readLine <<endl;**

**}**

**rules.close(); //closes file**

**//**

**// restart set to y so that you can enter while loop for the first time**

**restart="y";**

**// while statement used to loop again if user wants to do another round**

**while(answer!="N" && answer!="n"){**

**//**

**// if statement here if user wanted to restart game**

**if(restart=="y" || restart=="Y")**

**{**

**// set restart to nothing**

**restart="";**

**cout << "How many players are playing? Only 3 can play at the "**

**<< "same time plus the dealer." << endl;**

**cin >> numOfP;**

**//**

**//While statement tests user input for amount of players**

**while(numOfP<1 || numOfP>3){ // add more later so that it only accepts ints**

**if(numOfP==0)**

**{**

**cout <<"There's no one to play with! Try again." << endl;**

**cin>>numOfP;**

**}**

**else if(numOfP<0)**

**{**

**cout<<"Cannot enter negative numbers, try again."<<endl;**

**cin>>numOfP;**

**}**

**else if(numOfP>3)**

**{**

**cout <<"That's too many people! Try again." << endl;**

**cin>>numOfP;**

**}**

**}**

**//**

**// object for player needs to stay here in order to create an object**

**// each time the round restarts since the object is deleted after a**

**// reset**

**p = new Player[4]; // object array of 4 to hold 4 players**

**string name; // name used to enter new names in Player class**

**//**

**// Input name for players**

**cout << "Enter name for the players." << endl;**

**for(i=1;i<=numOfP;i++) //for loop used to enter name to each Player**

**{ //class array**

**cin >> name;**

**p[i].setName(name); //make an exception here**

**}**

**cout << endl;**

**p[0].setName("Dealer"); // Sets the name for the Dealer**

**//**

**// Put the default amount of credits for eacj player**

**for(i=1;i<numOfP+1;i++)**

**{**

**p[i]+=defCredit; // gives each plater 500 coin**

**}**

**// Statement to signify the first round**

**cout<<"OK! Round " << round << " starts now!"<<endl<<endl;**

**}//end of if statement**

**//**

**// Call the PlaceBets function so that we can place bets**

**put.PlaceBets(p,numOfP);**

**//**

**//Create hand array of objects to load in cards**

**load=new Hand[4];**

**int a=2,b=2,c=2,d=2; // gives the correct size of each hand**

**//**

**// for loop that draws cards for each player until there are 2 cards**

**for(i=0;i<=numOfP;i++)**

**{**

**for(j=0;j<2;j++)**

**{**

**load[i].setHand(deck,j);**

**}**

**}**

**//**

**cout<<endl;**

**//Output the contents of each players hand**

**if(numOfP>=1)**

**{**

**cout<<p[1].getName() << "'s hand" << endl;**

**cout<<p[1].getName() << "'s Coin : " <<p[1].getCredit()<< endl;**

**for(j=0;j<b;j++)**

**{**

**deck.PrintCard(load[1].getHand(j));**

**}**

**cout<<endl;**

**}**

**if(numOfP>=2)**

**{**

**cout<<endl<<p[2].getName() << "'s hand" << endl;**

**cout<<p[2].getName() << "'s Coin : " <<p[2].getCredit()<< endl;**

**for(j=0;j<c;j++)**

**{**

**deck.PrintCard(load[2].getHand(j));**

**}**

**cout<<endl;**

**}**

**if(numOfP>=3)**

**{**

**cout<<endl<<p[3].getName() << "'s hand" << endl;**

**cout<<p[3].getName() << "'s Coin : " <<p[3].getCredit()<< endl;**

**for(j=0;j<d;j++)**

**{**

**deck.PrintCard(load[3].getHand(j));**

**}**

**cout<<endl;**

**}**

**//**

**// Function call so that players can draw more cards**

**Dealer(deck,p,load,numOfP,a,b,c,d);**

**//**

**// Function call to check the sum and win conditions**

**CheckSum(p,load,numOfP,b,c,d);**

**CheckWin(p,put,numOfP);**

**//**

**cout<<"Round "<<round<<"Ended. Play another?"<<endl;**

**cout<<"'Y' for yes 'N' for no."<<endl;**

**cin>>answer;**

**// count round**

**round++;**

**//while statement used for input validation**

**while(answer!="N" && answer!="n" && answer!="Y" && answer!="y")**

**{**

**cout<<"Wrong input detected, only 'Y' for yes, or 'N' for no is"**

**<<"accepted. Try again."<<endl;**

**cin>>answer;**

**}**

**//**

**//delete allocated data here**

**cout<<"Would you like to reset game or continue?"<<endl;**

**cout<<"'Y' for yes 'N' for no."<<endl;**

**cin>>restart;**

**//while statement used for input validation**

**while(restart!="N" && restart!="n" && restart!="Y" && restart!="y")**

**{**

**cout<<"Wrong input detected, only 'Y' for yes, or 'N' for no is"**

**<<"accepted. Try again."<<endl;**

**cin>>restart;**

**}**

**if(restart=="y" || restart=="Y")**

**{**

**delete []p;**

**}**

**else if(restart=="n" || restart=="N")**

**{**

**cout<<"OK! Round " << round << " starts now!"<<endl<<endl;**

**}**

**deck.~Deck();**

**for(i=0;i<numOfP;i++)**

**{**

**load[i].~Hand();**

**}**

**}**

**cout<<"\n\nExiting Program"<<endl;**

**// Exit stage right**

**return 0;**

**}**

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

**\* Dealer Function:**

**\* This function allows player to draw additional cards from the deck.**

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

**void Dealer(Deck &deck, Player \*p, Hand \*l,int numOfP,int &a,**

**int &b, int &c, int &d)**

**{**

**int choice,i,j;**

**cout << "\nDo you want to draw more cards?"<<endl;**

**// if statement used to control what should be outputted**

**if (numOfP>=1)**

**{**

**cout << "Press 1 to let player one draw."<<endl;**

**}**

**if (numOfP>=2)**

**{**

**cout << "Press 2 to let player two draw."<<endl;**

**}**

**if (numOfP==3)**

**{**

**cout << "Press 3 to let player three draw."<<endl;**

**}**

**cout <<"Press 4 to continue."<<endl;**

**cin >> choice;**

**//**

**while(choice!=4){**

**//**

**if(choice==1)**

**{**

**l[1].setHand(deck, b);**

**b++;**

**}**

**else if(choice==2)**

**{**

**l[2].setHand(deck, c);**

**c++;**

**}**

**else if(choice==3)**

**{**

**l[3].setHand(deck, d);**

**d++;**

**}**

**else**

**{**

**cout<<"That is not a valid choice! Try again." << endl;**

**}**

**//**

**//Output the contents of each players hand with the if statement**

**if(numOfP>=1)**

**{**

**cout<<endl<<p[1].getName()<< "'s hand" << endl;**

**cout<<p[1].getName()<<" Coin : "<<p[1].getCredit()<<endl;**

**for(j=0;j<b;j++)**

**{**

**deck.PrintCard(l[1].getHand(j));**

**}**

**cout<<endl;**

**}**

**if(numOfP>=2)**

**{**

**cout<<endl<<p[2].getName() << "'s hand" << endl;**

**cout<<p[2].getName()<<" Coin : "<<p[2].getCredit()<<endl;**

**for(j=0;j<c;j++)**

**{**

**deck.PrintCard(l[2].getHand(j));**

**}**

**cout<<endl;**

**}**

**if(numOfP>=3)**

**{**

**cout<<endl<<p[3].getName() << "'s hand" << endl;**

**cout<<p[3].getName()<<" Coin : "<<p[3].getCredit()<<endl;**

**for(j=0;j<d;j++)**

**{**

**deck.PrintCard(l[3].getHand(j));**

**}**

**cout<<endl;**

**}**

**//**

**cout << "Do you want to draw more cards?"<<endl;**

**if (numOfP>=1)**

**{**

**cout << "Press 1 to let player one draw."<<endl;**

**}**

**if (numOfP>=2)**

**{**

**cout << "Press 2 to let player two draw."<<endl;**

**}**

**if (numOfP==3)**

**{**

**cout << "Press 3 to let player three draw."<<endl;**

**}**

**cout <<"Press 4 to continue."<<endl;**

**cin >> choice;**

**}**

**//**

**int total=0;**

**// calculates total for dealer**

**cout<<endl<<endl<<p[0].getName()<<endl;**

**// outputs the dealers hand**

**for(int i=0;i<a;i++){**

**deck.PrintCard(l[0].getHand(i));**

**p[0].setTotal(l,i);**

**}**

**//Maybe delete later**

**// p[0].total=total**

**while(p[0].getTotal()<17) // should draw until busts or wins**

**{**

**l[0].setHand(deck, a);**

**deck.PrintCard(l[0].getHand(a));**

**p[0].setTotal(l,a);**

**// check other version if this gives weird output**

**a++;**

**}**

**cout << p[0].getName() <<" has a value of " << p[0].getTotal() << endl;**

**}**

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

**\* CheckSum Function:**

**\* After the player(s) are done drawing cards form the Dealer function, this**

**\* function gets called to sum up the contents of each players hand.**

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

**void CheckSum(Player \*p,Hand \*l,int numOfP, int b, int c, int d)**

**{**

**int i;**

**// if statement**

**if(numOfP>=1)**

**{**

**for(i=0;i<b;i++)**

**{**

**p[1].setTotal(l,i);**

**}**

**cout <<endl<< p[1].getName() <<" has a value of " << p[1].getTotal()**

**<<endl<<endl;**

**}**

**if(numOfP>=2)**

**{**

**for(i=0;i<c;i++)**

**{**

**p[2].setTotal(l,i);**

**}**

**cout << p[2].getName() <<" has a value of " << p[2].getTotal()**

**<< endl<<endl;**

**}**

**if(numOfP>=3)**

**{**

**for(i=0;i<d;i++)**

**{**

**p[3].setTotal(l,i);**

**}**

**cout << p[3].getName() <<" has a value of " << p[3].getTotal()**

**<< endl<<endl;**

**}**

**}**

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

**\* CheckWin Function:**

**\* After the CheckSum function is called, this function gets called to see who**

**\* won. If sum is over 21, the function will say who busted.**

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

**void CheckWin(Player \*p,Bet &b , int numOfP)**

**{**

**//Variables to get max score**

**int max=p[0].getCredit();**

**int score;**

**string winner;**

**int j;//counter**

**for(int i=1;i<=numOfP;i++)**

**{**

**if(p[i].getTotal()<=21)**

**{**

**if(p[0].getTotal()<p[i].getTotal())// || p[0].getTotal()>21)**

**{**

**//if statement used to check every players score to see who won**

**//Checks for max total**

**if(max<p[i].getTotal())// fiqure something out lol**

**{**

**score=p[i].getTotal();**

**max=score;**

**winner=p[i].getName();**

**}**

**j++; //counts one loop**

**cout<<"This is the J counter... "<<j<<endl;**

**//if statement executes one j = numOfP**

**if(j=numOfP+1)**

**{**

**cout << winner <<" wins!" << endl;//player wins**

**cout<<"Your " << winner << "'s new score is "<<p[i].getCredit()<<endl;**

**p[i]+=(b.getLost(i))\*2;//make this so that it can get the losses**

**//of the player that bet**

**}**

**}**

**else if(p[0].getTotal()==p[i].getTotal())**

**{**

**cout << "Its a tie with the dealer with " << p[i].getName()**

**<<" and since its house rules, you lose your bet!"**

**<< " Play Again!"<<endl;//tie**

**cout <<p[i].getName()<< " lost $" << b.getLost(i)<<"."<<endl;**

**p[0]+=b.getLost(i);**

**}**

**else**

**{**

**cout <<"The dealer wins against "<<p[i].getName() << "."**

**<< endl;//Dealer Wins**

**cout <<p[i].getName()<< " lost $" << b.getLost(i)<<"."<<endl;**

**p[0]+=b.getLost(i);**

**}**

**}**

**else**

**{**

**cout<< p[i].getName() <<"'s hand busted! " <<p[i].getName()**

**<< " is out of the game."<< endl;**

**}**

**}//end of for loop**

**// Output**

**cout<<endl<<"Dealer's Coin :"<<endl;**

**cout<<p[0].getCredit()<<endl;**

**cout<<p[1].getName()<<"'s Coin :"<<endl;**

**cout<<p[1].getCredit()<<endl;**

**cout<<p[2].getName()<<"'s Coin :"<<endl;**

**cout<<p[2].getCredit()<<endl;**

**cout<<p[3].getName()<<"'s Coin :"<<endl;**

**cout<<p[3].getCredit()<<endl;**

**}**

**Hand.h**

**/\***

**\* File: Hand.h**

**\* Author: MSOS**

**\***

**\* Created on July 31, 2021, 1:52 PM**

**\*/**

**#ifndef HAND\_H**

**#define HAND\_H**

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

**\* Hand Class**

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

**class Hand**

**{**

**private:**

**Card hand[20];**

**public:**

**// Destructor**

**~Hand()**

**{**

**delete []hand;**

**}**

**// Accessor**

**void setHand(Deck &d, int i)**

**{**

**hand[i]=(d.draw());**

**}**

**//mutator**

**Card getHand(int x)**

**{**

**return hand[x];**

**}**

**};**

**#endif /\* HAND\_H \*/**

**Bet.h**

**/\***

**\* File: Bet.h**

**\* Author: MSOS**

**\***

**\* Created on July 28, 2021, 11:31 PM**

**\*/**

**#ifndef BET\_H**

**#define BET\_H**

**#include<string>**

**#include "Player.h"**

**using namespace std;**

**//class derived form player**

**//template <typename T>//do more on this in a**

**class Bet: public Player**

**{**

**private:**

**float bet; //ask how much the player wants to bet**

**float amount; //amount left over**

**float \*lost; //amount lost**

**public:**

**//default constructor**

**Bet()**

**{**

**bet=0;**

**amount=0;**

**}**

**//Paramerterized constructor**

**Bet(float b,float a) //poly**

**{**

**bet=b;**

**amount=a;**

**}**

**//destructor to erase bets if another new game is started**

**//Accessors(I dont think i need these...)**

**void setBet(float b)**

**{**

**bet=b;**

**}**

**void setAmount(float a)**

**{**

**amount=a;**

**}**

**//Mutators**

**float getBet()**

**{**

**return bet;**

**}**

**float getAmount()**

**{**

**return amount; //gives out new amount**

**}**

**float getLost(int index)**

**{**

**return lost[index];**

**}**

**//**

**// Member functions**

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

**\*PlaceBets Member Function:**

**\* This will ask player for**

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

**void PlaceBets(Player \*p, int numOP)**

**{**

**//Create array**

**lost=new float[numOP];**

**//Variables**

**string answer;**

**int pep;**

**//lists each players credits**

**for(int i=0;i<numOP+1;i++)**

**{**

**lost[i]=0; //sets all lost to zero**

**cout<<p[i].getName()<<endl;**

**cout<<p[i].getCredit()<<endl<<endl;**

**}**

**// Input**

**cout<<"Would you like to place bets? Type 'Y' for yes, 'N' for no."**

**<<endl;**

**cin>>answer;**

**// input validation**

**while(answer!="N" && answer!="n" && answer!="Y" && answer!="y")**

**{**

**cout<<"Wrong input detected, only 'Y' for yes, or 'N' for no is"**

**<<"accepted. Try again."<<endl;**

**cin>>answer;**

**}**

**//**

**// if statement which will exit class if answer was No**

**if(answer=="Y" || answer=="y")**

**{**

**answer="";//removes contents of answer.**

**//while statement used so that it loops each time the answer is**

**//yes**

**while(answer!="N" && answer!="n")**

**{**

**//resets pep to zero so that the while statement is used**

**pep=0;**

**//use if statement to take bet from correct player**

**if(numOP==1)**

**{**

**while(pep!=1)**

**{**

**cout<<"Press 1 if player 1 wants to give a bet."**

**<<endl;**

**cin>>pep;**

**}**

**}**

**else if(numOP==2)**

**{**

**while(pep!=1 && pep!=2)**

**{**

**cout<<"Press 1 if player 1 wants to give a bet."**

**<<endl<<"Press 2 if player 2 wants to give a"**

**<<" bet."<<endl;**

**cin>>pep;**

**}**

**}**

**else if(numOP==3)**

**{**

**while(pep!=1 && pep!=2 && pep!=3)**

**{**

**cout<<"Press 1 if player 1 wants to give a bet."**

**<<endl<<"Press 2 if player 2 wants to give a"**

**<<" bet."<<endl<<"Press 3 if player 3 wants to"**

**<<" give a bet."<<endl;**

**cin>>pep;**

**}**

**}**

**//**

**//if statement to go to correct player**

**if(pep==1)**

**{**

**cout<<"How much do you want to bet? You only have "**

**<<p[1].getCredit()<<endl;**

**cin>>bet;**

**// checks if the bet is within range**

**while(p[1].getCredit()<bet)**

**{**

**cout<<"Woah buddy, you're too broke to bet that "**

**<<"much. Bet below "<<p[1].getCredit()<<endl;**

**cin>>bet;**

**}**

**amount=p[1].getCredit();**

**p[1].setCredit(amount-bet);**

**cout<< p[1].getName() <<" new amount is "**

**<< p[1].getCredit() << endl << endl;**

**lost[1]+=bet;**

**}**

**else if(pep==2)**

**{**

**cout<<"How much do you want to bet? You only have "**

**<<p[2].getCredit()<<endl;**

**cin>>bet;**

**// checks if the bet is within range**

**while(p[2].getCredit()<bet)**

**{**

**cout<<"Woah buddy, you're too broke to bet that "**

**<<"much. Bet below "<<p[2].getCredit()<<endl;**

**cin>>bet;**

**}**

**amount=p[2].getCredit();**

**p[2].setCredit(amount-bet);**

**cout<< p[2].getName() <<" new amount is "**

**<< p[2].getCredit() << endl<<endl;**

**lost[2]+=bet;**

**}**

**else if(pep==3)**

**{**

**cout<<"How much do you want to bet? You only have "**

**<<p[3].getCredit()<<endl;**

**cin>>bet;**

**// checks if the bet is within range**

**while(p[3].getCredit()<bet)**

**{**

**cout<<"Woah buddy, you're too broke to bet that "**

**<<"much. Bet below "<<p[3].getCredit()<<endl;**

**cin>>bet;**

**}**

**amount=p[3].getCredit();**

**p[3].setCredit(amount-bet);**

**cout<< p[3].getName() <<" new amount is "**

**<<p[3].getCredit()<<endl << endl;**

**lost[3]+=bet;**

**}**

**//**

**cout<<"Will any others bet as well? Type 'Y' for yes, 'N' for"**

**<<" no."<<endl;**

**cin>>answer;**

**// nested while statement for user input**

**while(answer!="N" && answer!="n" && answer!="Y" && answer!="y")**

**{**

**cout<<"Wrong input detected, only 'Y' for yes, or 'N' for no is"**

**<<"accepted. Try again."<<endl;**

**cin>>answer;**

**}**

**}**

**}**

**else**

**{**

**cout<<"Continue on..."<<endl;**

**}**

**}**

**};**

**#endif /\* BET\_H \*/**

**Deck.h**

**/\***

**\* File: Deck.h**

**\* Author: MSOS**

**\***

**\* Created on July 27, 2021, 4:14 PM**

**\*/**

**#ifndef DECK\_H**

**#define DECK\_H**

**#include "Suits.h"**

**#include "CardName.h"**

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

**\* Deck Class**

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

**//inheritence and aggregation**

**class Deck:public Card**

**{**

**private:**

**Card arrCards[52]; // make into a dynamically allocated structure array**

**int index;**

**static int place;**

**public:**

**// Constructor**

**Deck(){**

**index=0;**

**for(int col=(int)Suits::Clubs; col<=(int)Suits::Spades; col++)**

**{**

**for(int row=(int)CardName::Ace; row <= (int)CardName::King; row++)**

**{**

**Card c;**

**c.suit = (Suits)col; // same as Suits[col]**

**c.name = (CardName)row; // these two access the enum class**

**// This gives each name its value**

**if (c.name==CardName::Jack){**

**c.value=10;**

**}**

**else if(c.name==CardName::Queen){**

**c.value=10;**

**}**

**else if(c.name==CardName::King){**

**c.value=10;**

**}**

**else**

**{**

**c.value = (int)c.name;**

**}**

**arrCards[index++]=c;**

**}**

**}**

**index--;**

**this->shuffle(); //call shuffle function**

**}**

**//destructor(probably will not use, try it out by calling it at the end or**

**// maybe it could be used to reshuffle the deck)**

**// ~Deck(void){**

**// delete []arrCards;**

**// }**

**//accessor Function**

**void setArrayOfCards(Card c){**

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

**arrCards[i]=c;**

**}**

**}**

**//mutator Function**

**Card getArrayOfCards() const{**

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

**return arrCards[i];**

**}**

**}**

**//member functions (methods)**

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

**\* PrintCard Function:**

**\* Will print out the card that is called.**

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

**void PrintCard(Card c)**

**{**

**PrintValue(c);**

**cout << " of ";**

**PrintSuit(c);**

**cout << endl;**

**}**

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

**\* PrintValue Function:**

**\* Prints out the value**

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

**void PrintValue(Card c)**

**{**

**if(c.name==CardName::Ace)**

**{**

**cout<<"Ace";**

**c.value=11;**

**}**

**else if(c.name==CardName::Jack)**

**{**

**cout<<"Jack";**

**c.value=10;**

**}**

**else if(c.name==CardName::Queen)**

**{**

**cout<<"Queen";**

**c.value=10;**

**}**

**else if(c.name==CardName::King)**

**{**

**cout<<"King";**

**c.value=10;**

**}**

**else**

**{**

**cout << c.value;**

**}**

**}**

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

**\* PrintSuit Function:**

**\* Prints out the suit to each respective card**

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

**void PrintSuit(Card c)**

**{**

**if (c.suit==Suits::Clubs)**

**{**

**cout << "clubs";**

**}**

**else if(c.suit==Suits::Diamonds)**

**{**

**cout << "diamonds";**

**}**

**else if(c.suit==Suits::Hearts)**

**{**

**cout << "hearts";**

**}**

**else if(c.suit==Suits::Spades)**

**{**

**cout << "spades";**

**}**

**}**

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

**\* Shuffle Function:**

**\* Will shuffle the deck of cards**

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

**void shuffle()**

**{**

**int n,m;**

**Card temp=arrCards[0]; // makes temp start at the first Card struct**

**for (n=0;n<52;n++)**

**{**

**m=rand()%52; // gives a random number**

**temp=arrCards[n]; // switches the contents of d.arrCards[n] with what**

**// is in d.arrCards[m]**

**arrCards[n]=arrCards[m];**

**arrCards[m]=temp;**

**}**

**}**

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

**\* PrintAll Function:**

**\* This function will print the cards of the deck(just a test case)**

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

**void PrintAll()**

**{**

**for(int col=(int)Suits::Clubs; col<=(int)Suits::Spades; col++)**

**{**

**for(int row=(int)CardName::Ace; row <= (int)CardName::King; row++)**

**{**

**int index=(13\*col)+row-1; // accesses the contents of arrCards**

**PrintCard(arrCards[index]); // Prints contents in a card struct**

**}**

**}this->index; //**

**//PrintCard(d.arrCards[0]); // this can call each individually**

**}**

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

**\* Draw Structure Function:**

**\* This function allows to draw a card from the deck of cards**

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

**Card draw()**

**{**

**return arrCards[index--]; //returns whole struct array**

**}**

**};**

**#endif /\* DECK\_H \*/**

**Player.h**

**/\***

**\* File: Player.h**

**\* Author: MSOS**

**\***

**\* Created on July 27, 2021, 4:15 PM**

**\*/**

**#ifndef PLAYER\_H**

**#define PLAYER\_H**

**#include "Deck.h"**

**#include "Card.h"**

**#include "Hand.h"**

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

**\* Player Class**

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

**//template <typename T>**

**class Player**

**{**

**protected:**

**string name;**

**int total;**

**float credit;**

**public:**

**//defualt constructor**

**Player()**

**{**

**name="";**

**total=0;**

**credit=0;**

**}**

**// constructor**

**Player(string n){//poly??**

**name=n;**

**total=0;**

**credit=0;**

**}**

**//accessor**

**void setName(string n)**

**{**

**name=n;**

**}**

**void setTotal(Hand \*l,int x)**

**{**

**total+=l->getHand(x).value;**

**}**

**void setCredit(float c)**

**{**

**credit=c;**

**}**

**//mutator**

**string getName()**

**{**

**return name;**

**}**

**int getTotal()**

**{**

**return total;**

**}**

**float getCredit()**

**{**

**return credit;**

**}**

**friend Player operator+=(Player &p,float c){ //operator overloading+Freind thing**

**p.credit+=c;**

**return p;**

**}**

**// Member functions**

**};**

**#endif /\* PLAYER\_H \*/**

**Card.h**

**/\***

**\* File: Card.h**

**\* Author: MSOS**

**\***

**\* Created on July 27, 2021, 4:14 PM**

**\*/**

**#ifndef CARD\_H**

**#define CARD\_H**

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

**\*Card Class**

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

**class Card {**

**public: //need this to do the inheritance req**

**CardName name;**

**Suits suit;**

**int value;**

**//member functions**

**};**

**#endif /\* CARD\_H \*/**

**#ifndef CARDNAME\_H**

**#define CARDNAME\_H**

**enum class CardName{**

**Ace=1, // makes ace start at one in the enum**

**Two,**

**Three,**

**Four,**

**Five,**

**Six,**

**Seven,**

**Eight,**

**Nine,**

**Ten,**

**Jack,**

**Queen,**

**King**

**};**

**#endif /\* CARDNAME\_H \*/**

**Suits.h**

**#ifndef SUITS\_H**

**#define SUITS\_H**

**enum class Suits{**

**Clubs,**

**Diamonds,**

**Hearts,**

**Spades,**

**};**

**#endif /\* SUITS\_H \*/**