**Project 1**

Title:

**David’s**

**21 OR BUST**

**Blackjack**

Course:

**Fall CIS-18**

Section:

**47914**

Due Date:

**October 31, 2018**

Author:

**David Fernandez**

Table of Contents

[1. Introduction 4](#Introduction)

[2. Rules 4](#Rules)

[3. Gameplay 5](#Gameplay)

[4. Development Summary 6](#Development)

[4.1 Code Breakdown 6](#Breakdown)

[5. Specifications 9](#Specs)

[5.1 Sample Input/Outputs 9](#Sample)

[5.2 Flow Charts 12](#Flow)

[6. References 16](#Refrences)

[7. Program code 17](#Program)

1. Introduction

Even though many people are familiar with the basic rules of the game, only few are knowledgeable in the odds. Looking at the game from the outside deceives many people into believing that blackjack is a simple game causing them to lack the right strategy and knowledge. Causing many players fail to understand how complex blackjack actually is. Despite being one of the most complex casino games of all times, it is also one of the most popular ones.

**Blackjack is one of the most played casino games of all time** and also known as twenty-one (21) to many other players due to the importance of the number in the game’s objective to win. It is predominately offered by almost every casino all around the world for its popularity and simplicity. Which creates a great convenience for those who are fans and enjoy the game and also enjoy traveling. So far, **the game’s origin remains unknown** and the first written evidence of it was found in the 17th century.

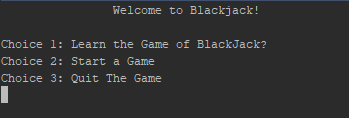
Blackjacks popularity grew greatly such as other casino known games. Much attributed to the simplicity of the game and the belief that it can be won in your favor with the right strategy and knowledge of the game. Yet, with this simplicity and ability to have an advantage there comes a great cost of preparation and knowledge compared to other games, but players can be sure that all of their efforts and time will be worthwhile. The more you study and practice the way to gain advantage of the game, the better chances they have of beating the casino, house, or dealer. This is the reason why there are a number of different strategies, they can learn and try which makes the whole game extremely interesting and challenging.

2. Rules

Blackjack may be played with multiple decks of 52-card decks. Aces are be counted as a 1 or 11 points, 2 to 10 according to the face value, and face cards (K, Q, J) count as ten points. The value of a hand is the sum of the total point values of the individual cards in your hand. A blackjack is the highest hand winning hand a player can achieve consisting of an ace and any 10-point card, and it outranks all other 21-point hands. After all the players have staked their bets limited by the amount given with a starting amount in the game being for this case equal to one hundred, the dealer will draw two cards to each player and then two cards to himself. One of the dealer cards is dealt face up and his other card is to be dealt face down. The face down card is called the hole card. If the dealer has a blackjack, then all wagers will lose but, in these games sense it will be pushed if any of the other players have a sum total equal to 21. After each player has had his turn, the dealer will turn over his hole card. If the dealer has 16 or less, then he will draw another card. A special situation is when the dealer has an ace and any number of cards totaling six points (known as a 'soft 17'). A soft hand is any hand that contains an ace and the ace’s value can still be either 11 or 1. If the dealer goes over 21 points, then any player who hadn’t already bust will win. If the dealer does not bust, then the higher point total between the player and dealer will win.

During each turn the player has multiple option to pick from. He has the opportunity to ‘Hit’ or ‘Stand’ on his current cards face value total. Hitting is when a player draws another card (and more if he wishes) but, if this card causes the player's total points to exceed 21 (known as 'breaking' or 'busting') then he loses. The game despite the dealer’s hand. On the other hand, is ‘Standing’ which is when the total value of your hand is 21 or lower allowing the opportunity of you choosing to not take anymore cards and not to risk the chance of your hand(s) exceeding 21 in total value.

3. Gameplay

The Game begins on a start up screen welcoming you to the game of blackjack. It then prompts you to select on of the

options available. By pressing 2 and then followed by enter you will be taken to the game screen in which you will play blackjack with friends or alone with starting cash of 100 dollars against a programed dealer looking to suck you dry. Each hand you will be allowed to play with or without having to bet against the dealer. There is a percent ‘bust’ tip given to each player at the start of their turn allowing them to think if they should take the risk knowing one dealer card or stand. Use your initial gamble amount of 100 dollars and amass yourself a fortune in this Blackjack console simulator “David’s 21 OR BUST Blackjack”.

4. Development Summary

|  |  |
| --- | --- |
| Lines of Code | 752 |
| Comment Lines | 106 |
| Blank Lines | 97 |
| Total Source Lines | 955 |
|  |  |

The project is a C++ Object Oriented Program of the casino game Blackjack. It was coded in the IDE Netbeans IDE 8.2. Using the Cygwin collection as its build tools. The project consists of 9 source files (.ccp) and 8 header files (.h).

This Project took a total of an estimated 20 hours of research, pseudo coding, fixing errors. Not including write-up and flowcharting.

4.1 Code Breakdown

**Card**

The card class creates the face values and suits of the potential cards. The faces of the cards value ‘Rank’ is an enum that goes from Ace-King. The suits of the card are also an enum of the 4 possible suits. Creates and returns the ‘rank’ or value of the card with two if statements. First being if the card is faced up or not then if the value of the card is in the bounds. Also, Card gives the state of the cards position (faced-up or faced-down) with Flip().

**Hand**

The hand function inherits the card class and controls the addition of a card to the deck using simple push\_back. It also is in charge of clearing the memory on the heap and all vector pointers. Lastly, it returns the value of the hand total, the sum of all the cards, including the logic of an Ace being a 1 if not busting or a 1 if 11 causes a bust. This is done through a bool, for loops, and if statements.

**PlayerInfo**

PlayerInfo inherits the Hand class and manages the system of if the player has busted or gotten a 21. Using Booleans to state if either are true and if so calls its other function to notify the player of either losing or congratulate them on getting a 21. Lastly, it is where the name of the player is stored and managed.

**Deck**

The Deck class inherits the Hand and PlayerInfo classes and creates a deck. First it uses the vector reserves to create a Card array size equal to that of a standard 52 card deck then it populates it using nested for loops to create a perfect 52 card deck of unique cards with help from the Hand add function. It then uses the random\_shuffle function to randomize the deck that was created. This class manages the deal of the cards to the players personal hand as well as the dealers. Also, it manages hitting a card in a while loop as long as you don’t bust, hit 21, or decide to stand. Lastly, it also provides a tip to the players on their first hit option giving them the percent of busting from hands 12-20.

**Player**

The Player class is a child class to PlayerInfo and is in charge of many functions of the players outcome. The Player class manages the logic of determining if the player is wanting to Hit or Stand to relay to the deck class. The player class manages the betting system of all players initializing a total for all of 100. The Player class controls the players total cash amount and deducts or adds using simple logical operators and then retells the player of his current total amount. The class also makes certain the player can not trade more than he has and validates all wild entries for each question asked within itself.

**House**

The House like the player class is a child class to the PlayerInfo class. It’s duties are to Flip the hidden card once all players have finished their turn using the card->flip() function. Also, it manages the Houses hitting Boolean of if its under 16 hit is true and if above hit is false.

**Rules**

The Rules Class has no inheritance since it is simply a menu using a switch function prompting the users of a menu for help in knowing the game. There is 4 option in the switch(1:Rules of Blackjack 2:What is ‘Hit’ 3: What is ‘Stand’ 4: Return to Main Menu). The first three options Display a text description of the meaning or rules that are associated with the choice. The fourth simply takes you back to the main menu.

**Game**

The Game class inherits the Player, Deck, and House classes and is where the game is actually put together. It creates instances of a Deck, Player, and House variable. Game starts at the constructor By creating a vector of names and creating the rand seed. Then populates and shuffles a deck. The play() function is where the game is presented it clears the console and asks all players if they are betting. It uses an iterator for player and with a for loop it places all the bets of the player if they decided on it. It then creates the game scenario by using a nested for loop with the iterator to generate 2 cards while also generating the single house card and declaring the first card faced-down. After the cards have been displayed You are prompted if you would like to hit with the percentage of busting using the created Deck.BustingTip(pPlayer->GetTotal()); Deck.AdditionalCards(\*pPlayer); functions in another iterator for loop. After all players decide they are fine with their hands or have busted the House calls to see if it would like to hit or not. After that using nested if statements and for loops it determines who wins or if a push is necessary. While also adjusting the totals of the players who betted on the hands with either pPlayer->Win() or p Player->Lose(). Finally, it clears the hands of the individuals and house.

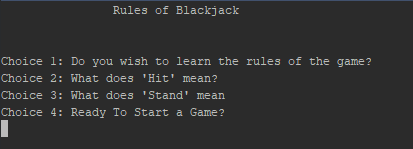
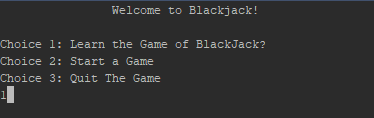
**Main**

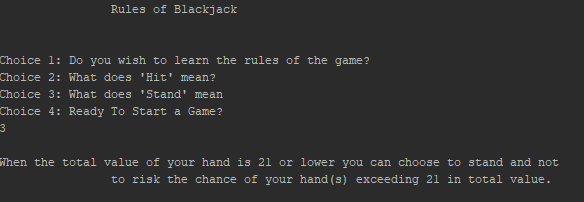
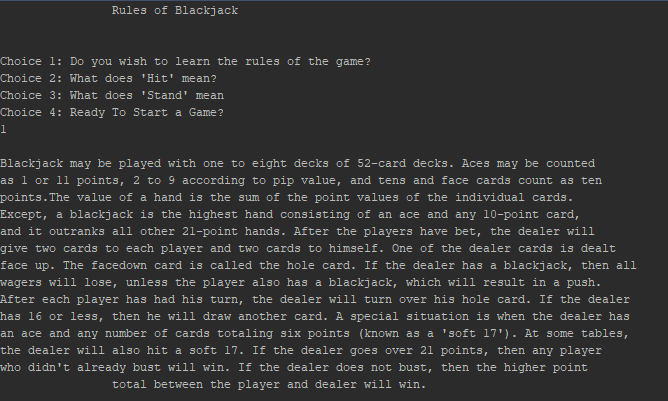
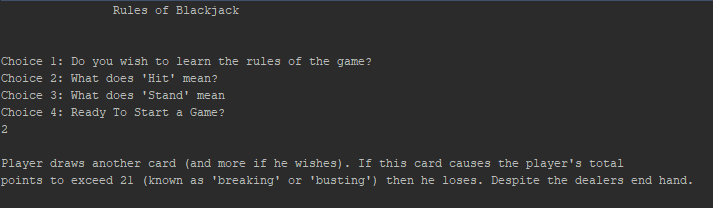
The main has no header and simply inherits the Game class. The Main has 3 function prototypes 2 overload operators so that Card object and PlayerInfo Object can be sent to cout using ostream (ostream& operator<<(ostream& os, const Card& aCard), ostream& operator<<(ostream& os, const PlayerInfo& aPlayerInfo)). The last prototype is the void start() function. The main method in the class Presents us with a welcoming menu screen presented earlier with 3 choices set on a switch and validated to only allow the 3 inputs or they would be presented with an error message. The First choice send you to the Rule class and shows you the rule menu. The second option is the start() function where the game is begun. In the start method you are asked for the number of players ranging from 1-7 and not allowing any other input or they will be prompted by an error and repeat message. The names of which these players would like to be addressed as no restrictions to be stored in a string vector. After all that an instance game called aGame is created after the game is finished you have the opportunity to either continue being dealt hands or to simply leave to the menu and quit or reread the rules.

**5. Specifications**

**5.1 Sample Input/Output**

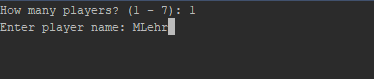
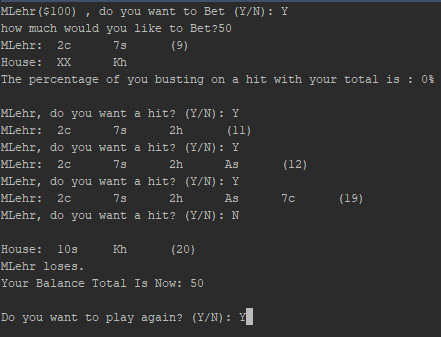
First, I will demonstrate the range of choice 1 from the main menu.

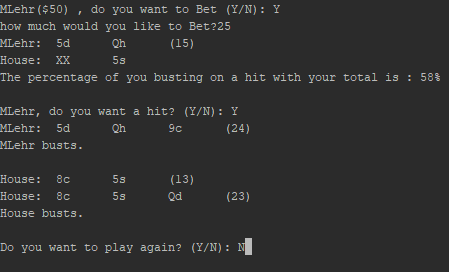
Which we know to be the RulesMenu and all that’s resides in it.

****

Choice 4 simply returns you to the main menu seen in Figure 1. Now I will provide the screens of the game in progress.

First you will be prompted of the number of players and their respected names

Then you will be prompted on whether you would like to bet or not and the amount. Afterwards the game begins and you will be able to play the game as long as you please.



With each new game the cash does follow and might just dry up but don’t worry you can always start over.

**5.2 Flow Charts**

**Card.Cpp**

Function GetValue() Line 6

**Value = 10**

True

**Value>10**

**Function Protoypes:**

None

**Global Variables:**

None

**Author:** David Fernandez

**Flow Chart Date:** October 30, 2018

**Purpose:** Create a Card

Nested If statements

**System Libraries:**

I/O Objects

Standard Namespace

Card.h

Card.h

**Declare Variable**

**Value**

**IfFaceUp**

False

**Value = Rank**

False

True

**Main.Cpp**

**B**

Start()

Main()

Rules.RulesMenu()

Main()

Exit

Start Game

Rules

Menu

**A**

Output the Menu

Input the choice

**Declare Variables**

choice

Rule

**Function Protoypes:**

ostream& operator<<(ostream& os, const Card& aCard);

ostream& operator<<(ostream& os, const PlayerInfo& aPlayerInfo);

void start();

**Global Variables:**

None

**System Libraries:**

Game.h

**Author:** David Fernandez

**Flow Chart Date:** October 31, 2018

**Purpose:** Main Menu Selection

Function main() Line 7

Switch

True

True

True

True

**A**

Main()

Error Msg

Choice >=’1’ &&

Choice <=’3’

**B**

**House.Cpp**

**Global Variables:**

None

**System Libraries:**

House.h

**Author:** David Fernandez

**Flow Chart Date:** October 31, 2018

**Purpose:** Flip the Dealer’s First Card

Function FlipFirstCard() Line 15

If-Else Statement

!Card.Empty

True

Card[]->Flip()

"No card to flip!"

**Function Protoypes:**

None

6. References

Sites provided in previous Lab Assignment:

<http://www.josuttis.com/libbook/algolist.pdf>

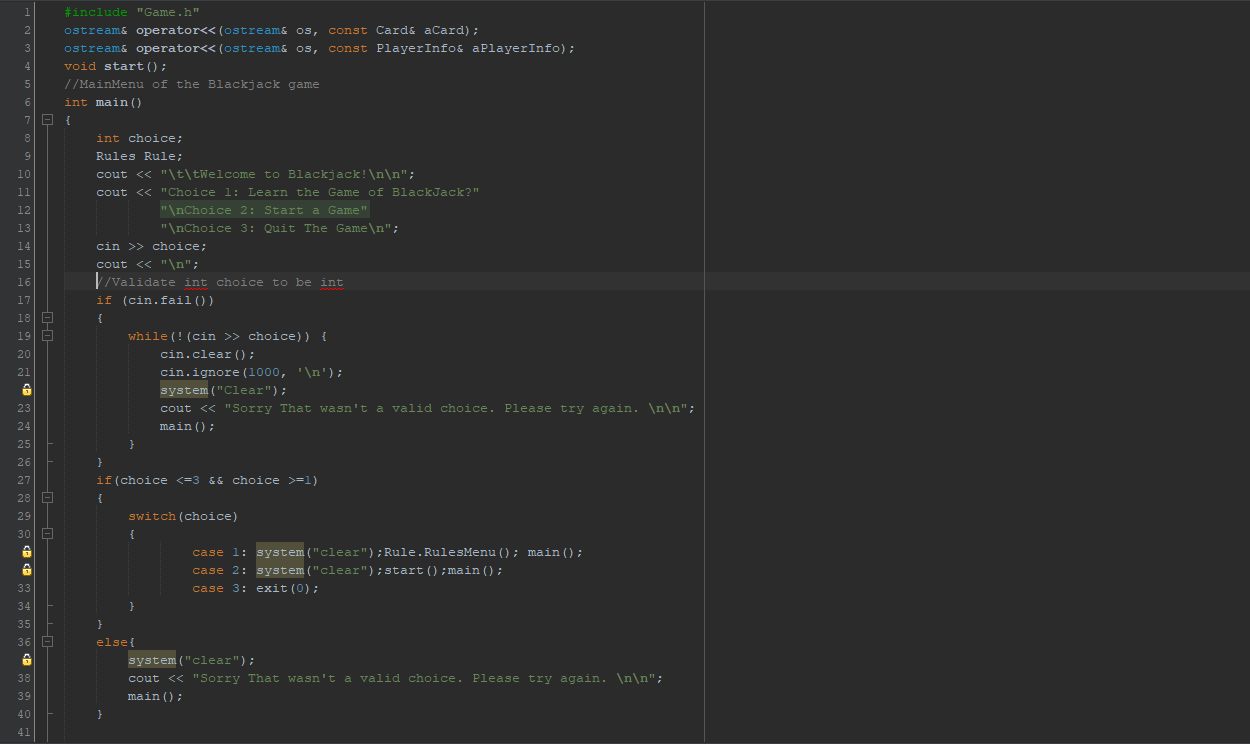
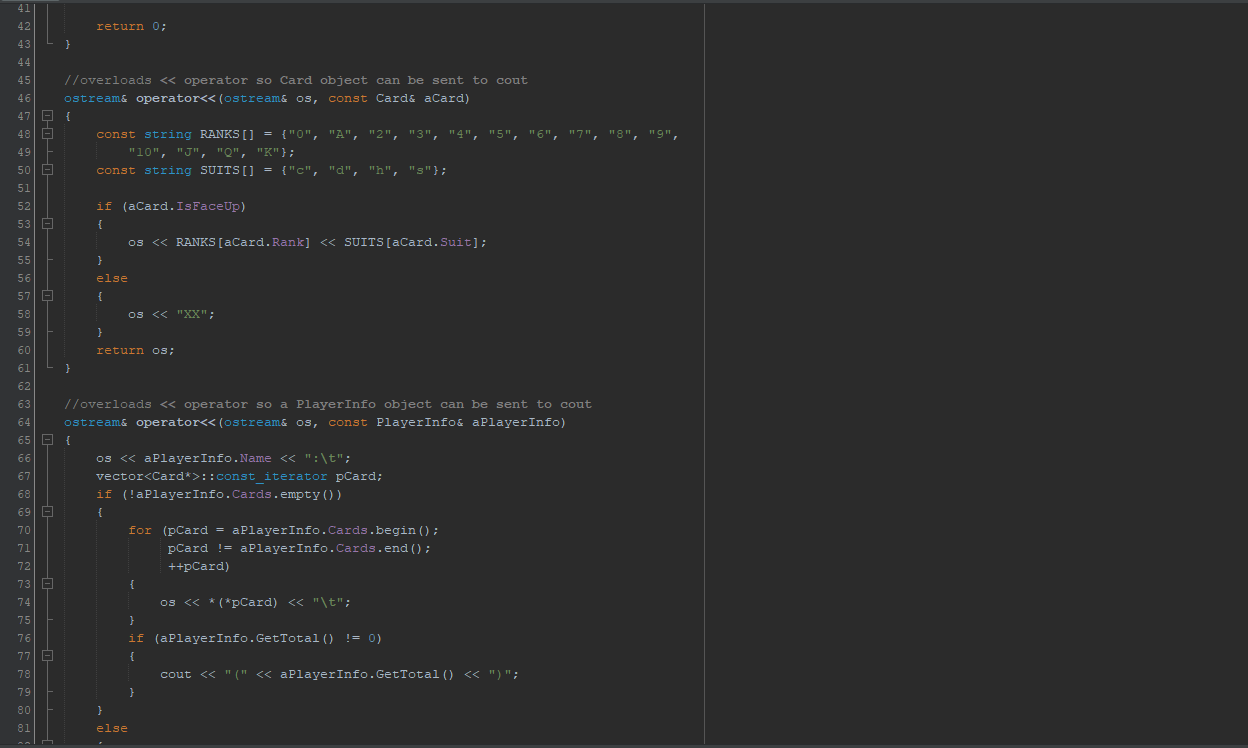
<http://www.tmplbook.com/>

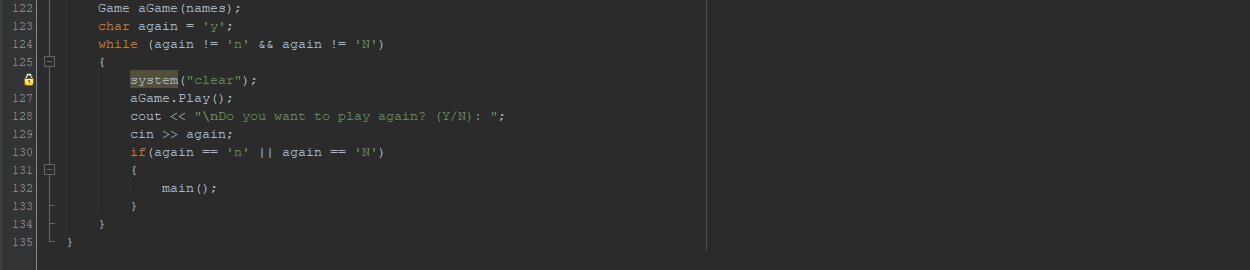
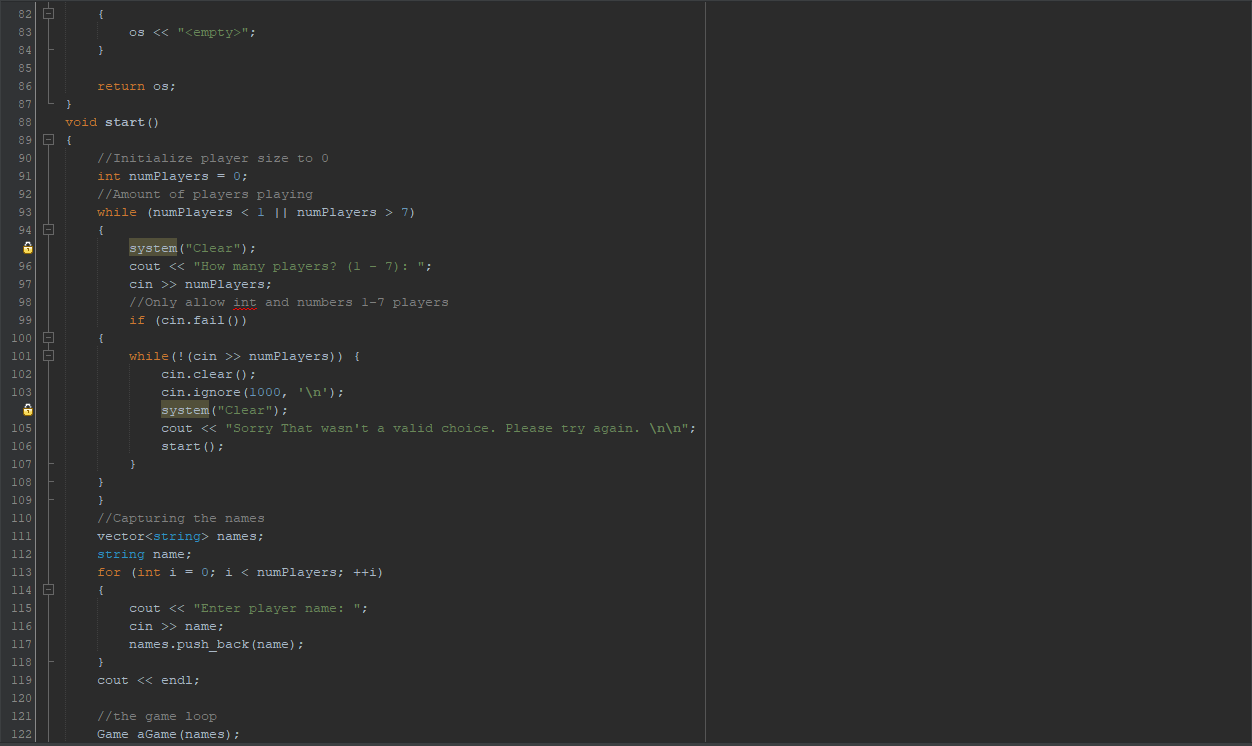
For Official Rules of Blackjack:

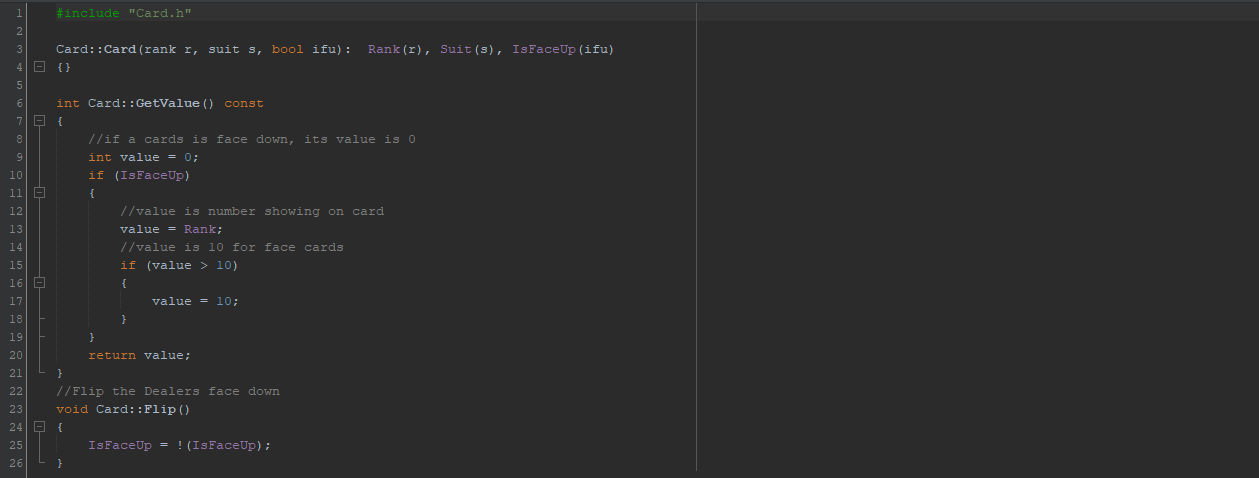
<https://www.bicyclecards.com>

For Others I used google and researched any ideas of solutions to any errors I had or correct my code. Code was not stolen merely used as tools of research and tool for learning in future works.

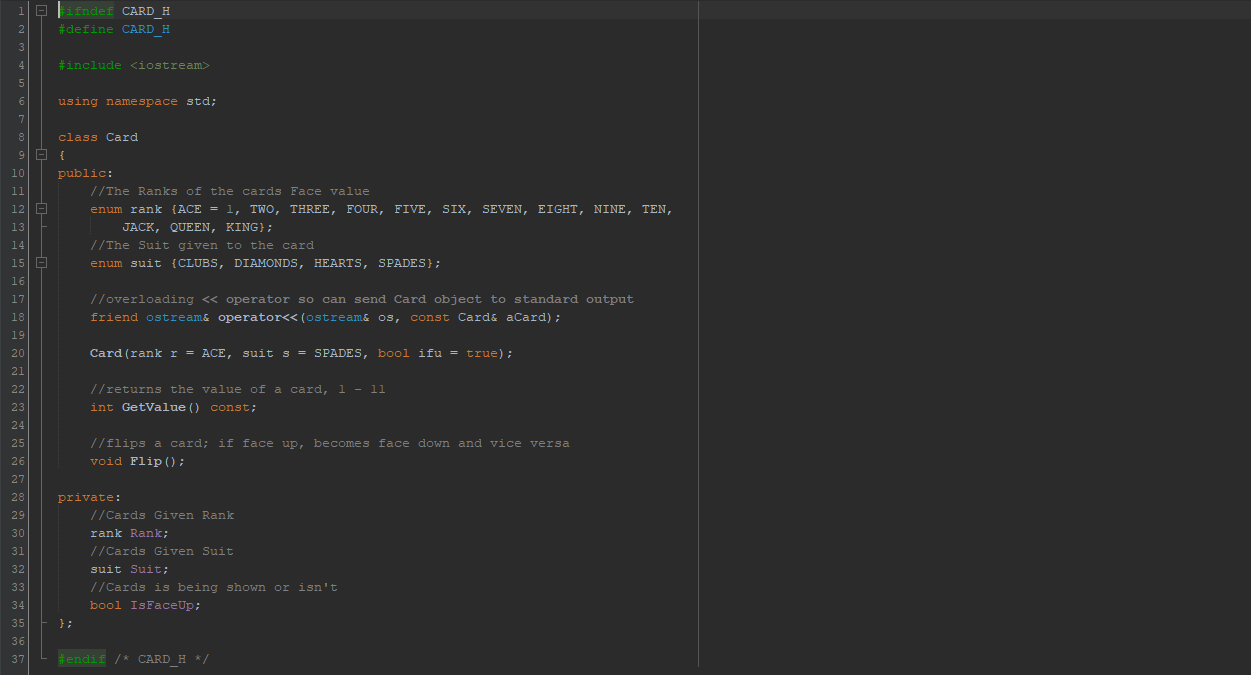
7. Program Code

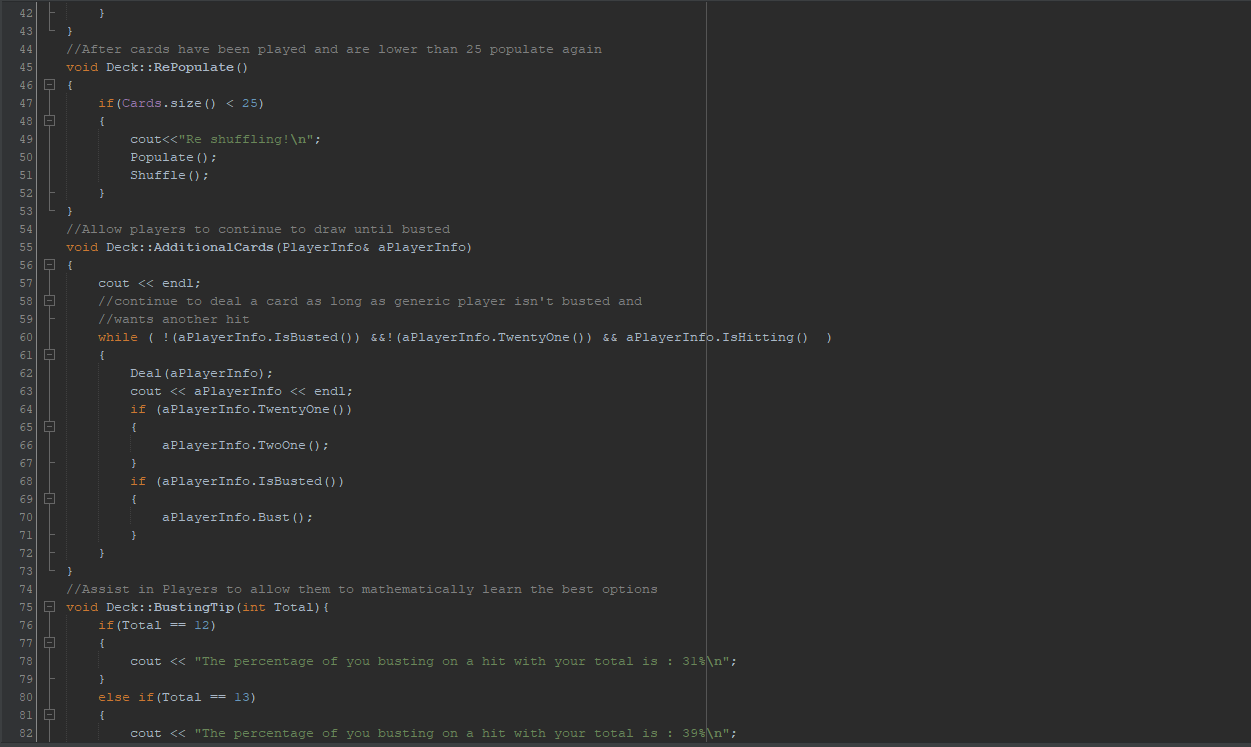
**Main.Cpp**

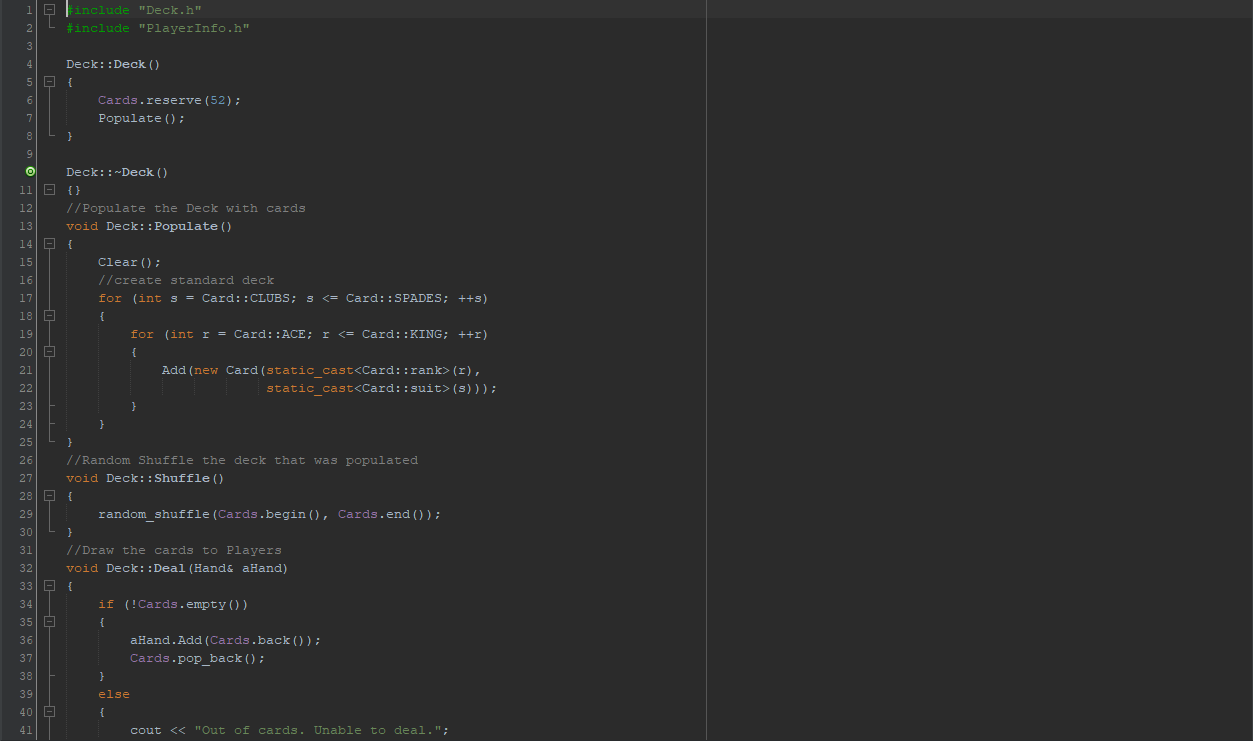
****

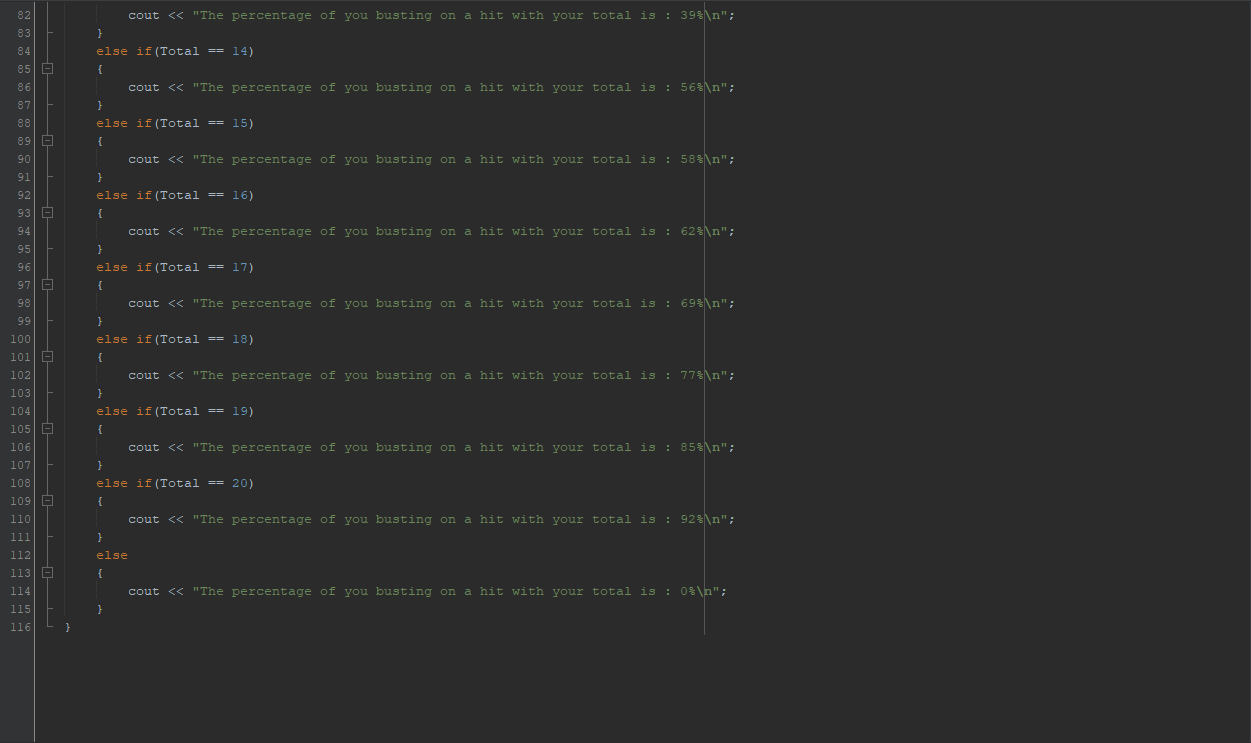
**Card.Cpp**

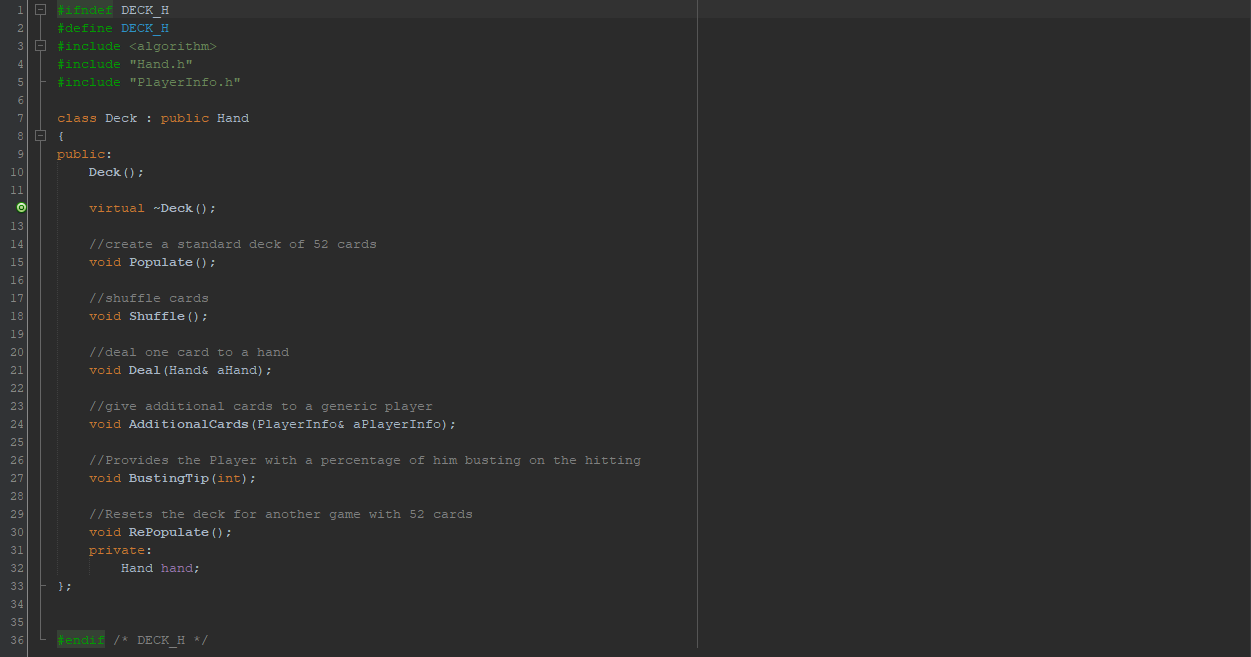
**Card.h**



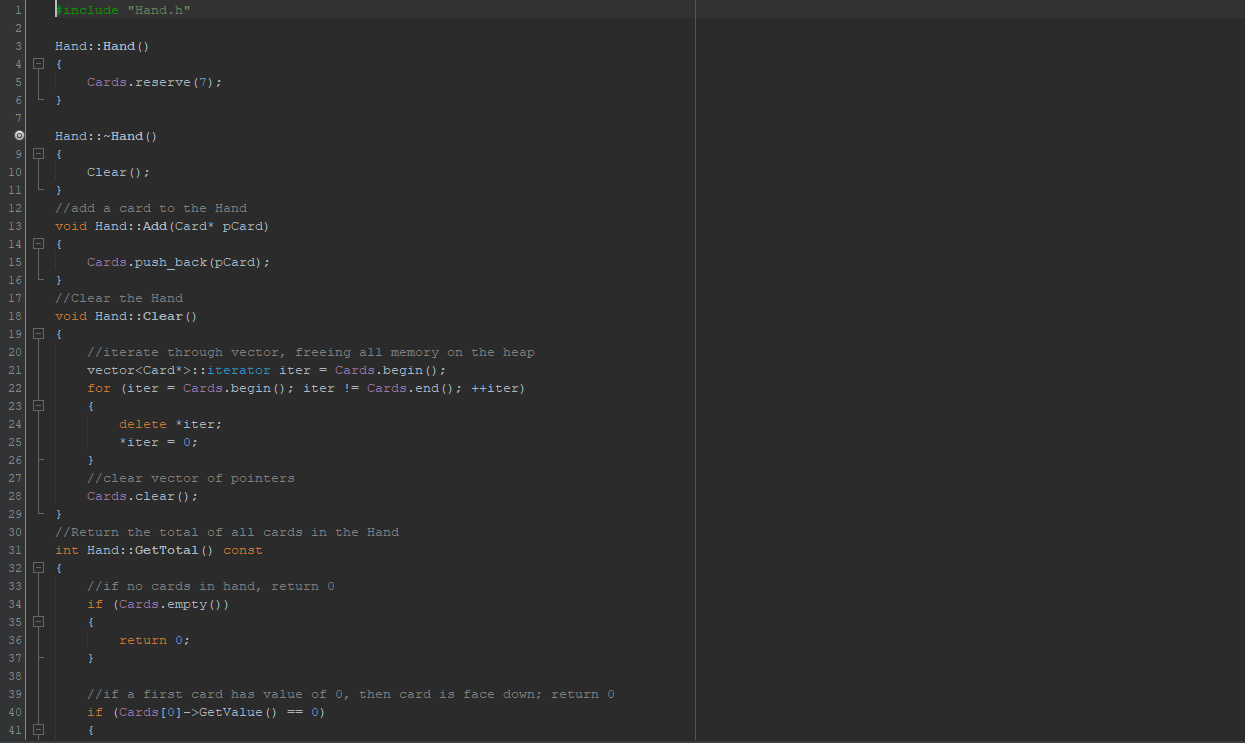
**Deck.cpp**

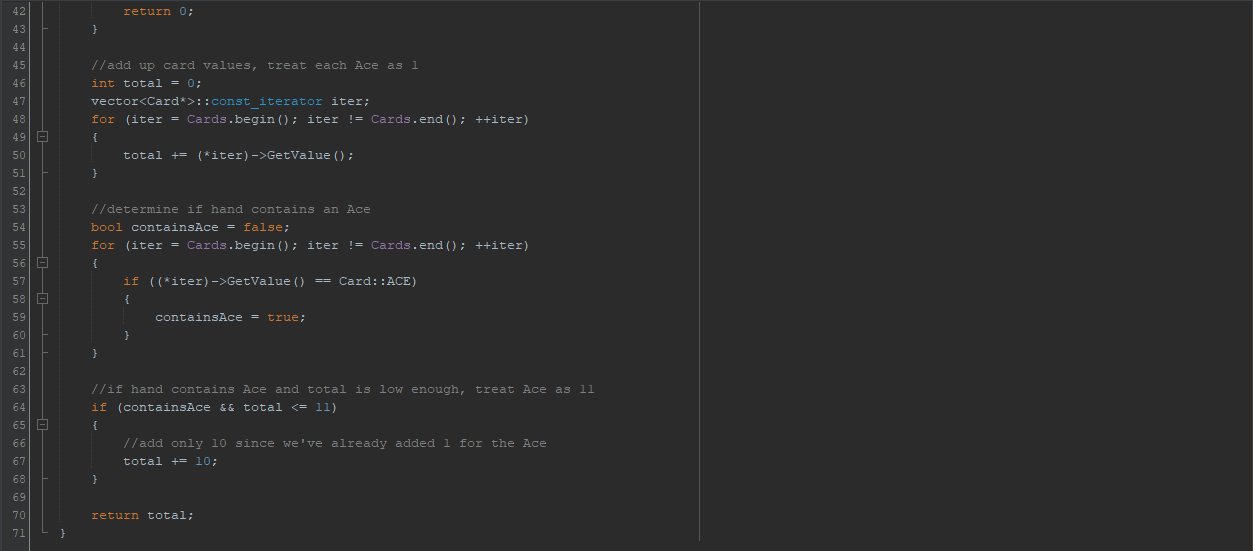


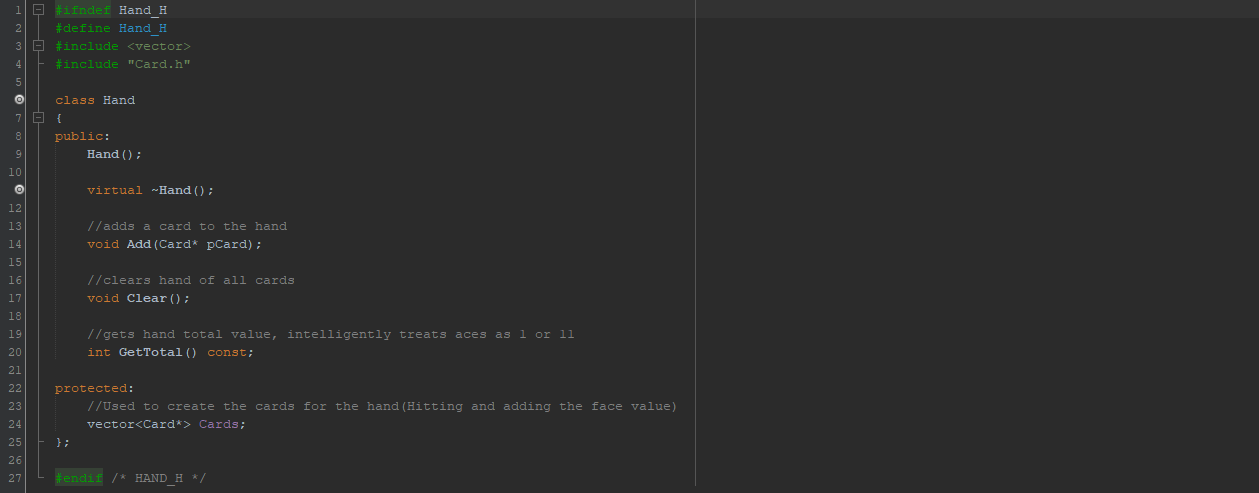


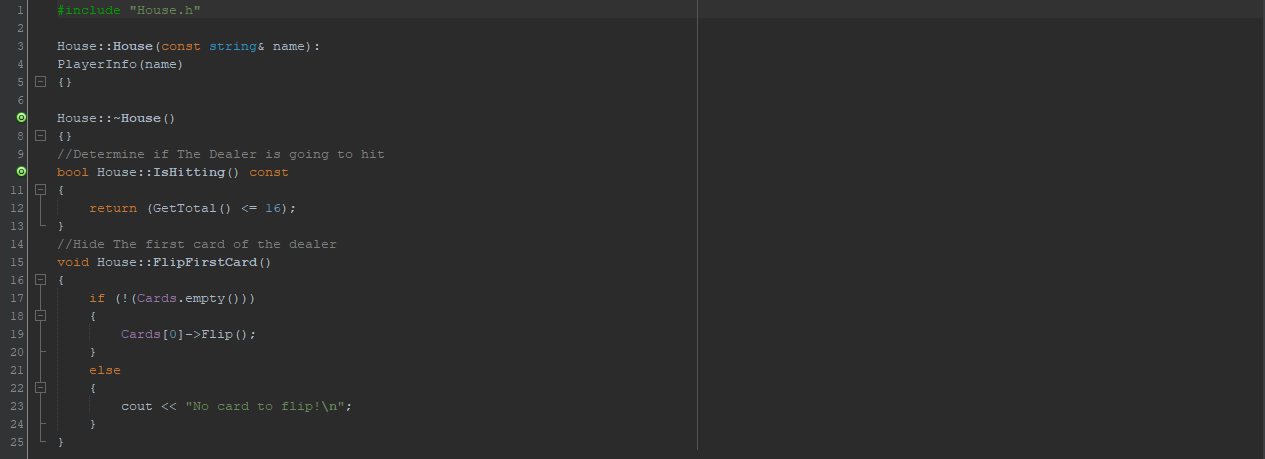
**Deck.h**

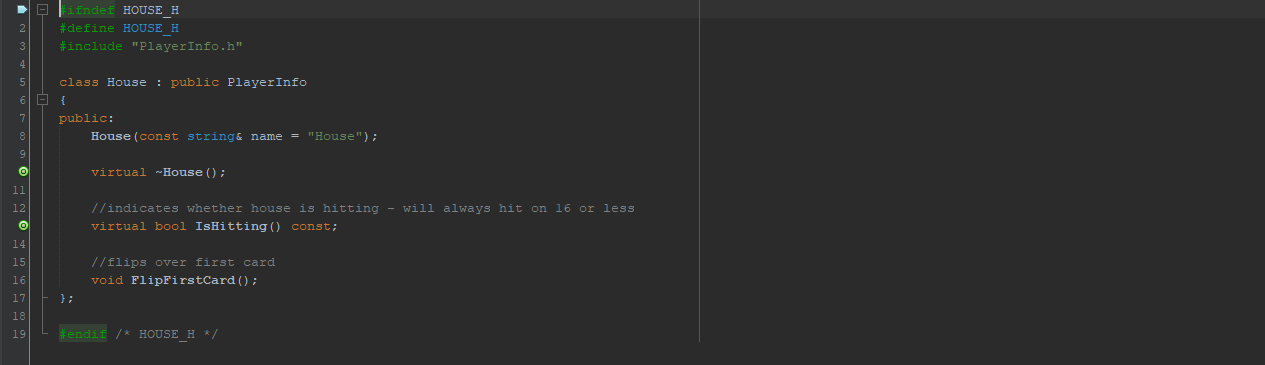
**Hand.cpp**

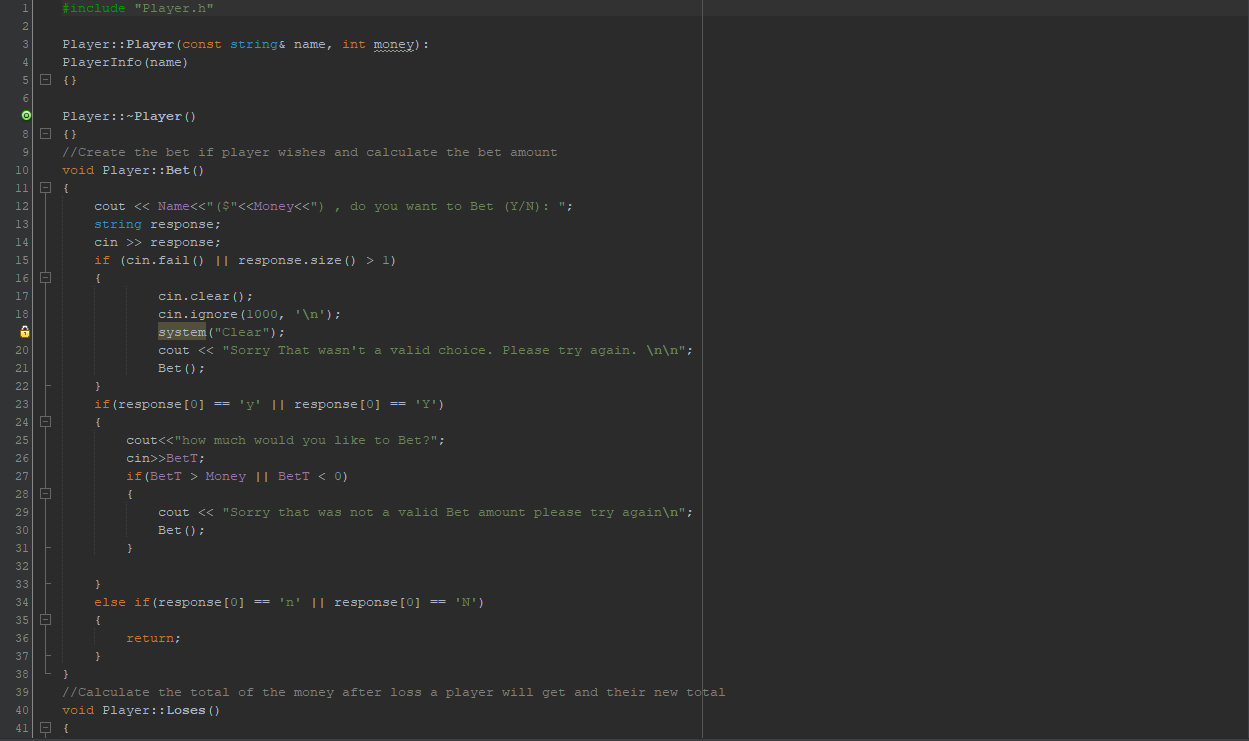


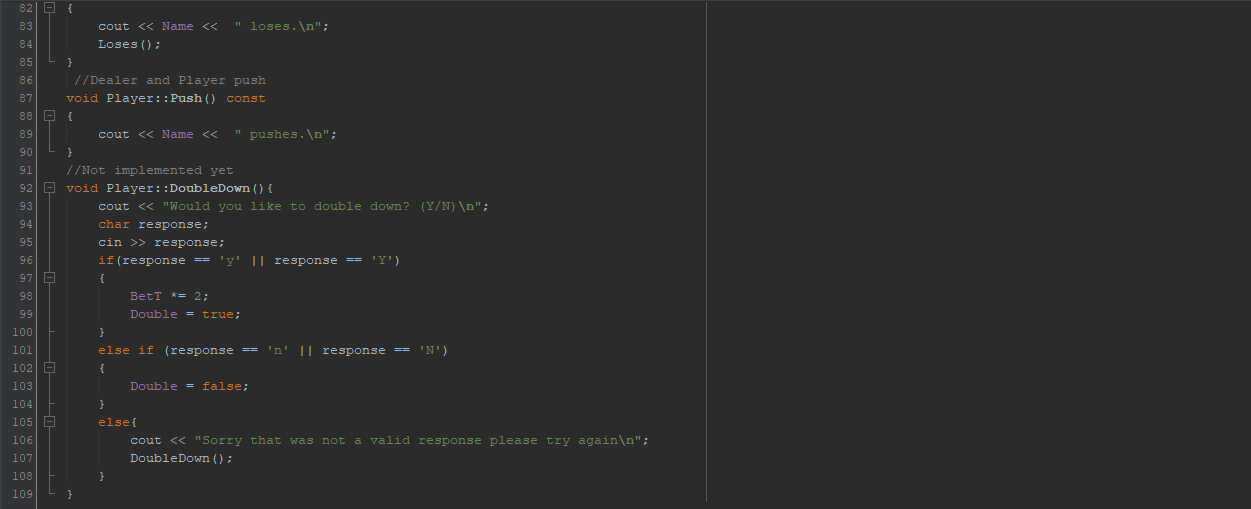
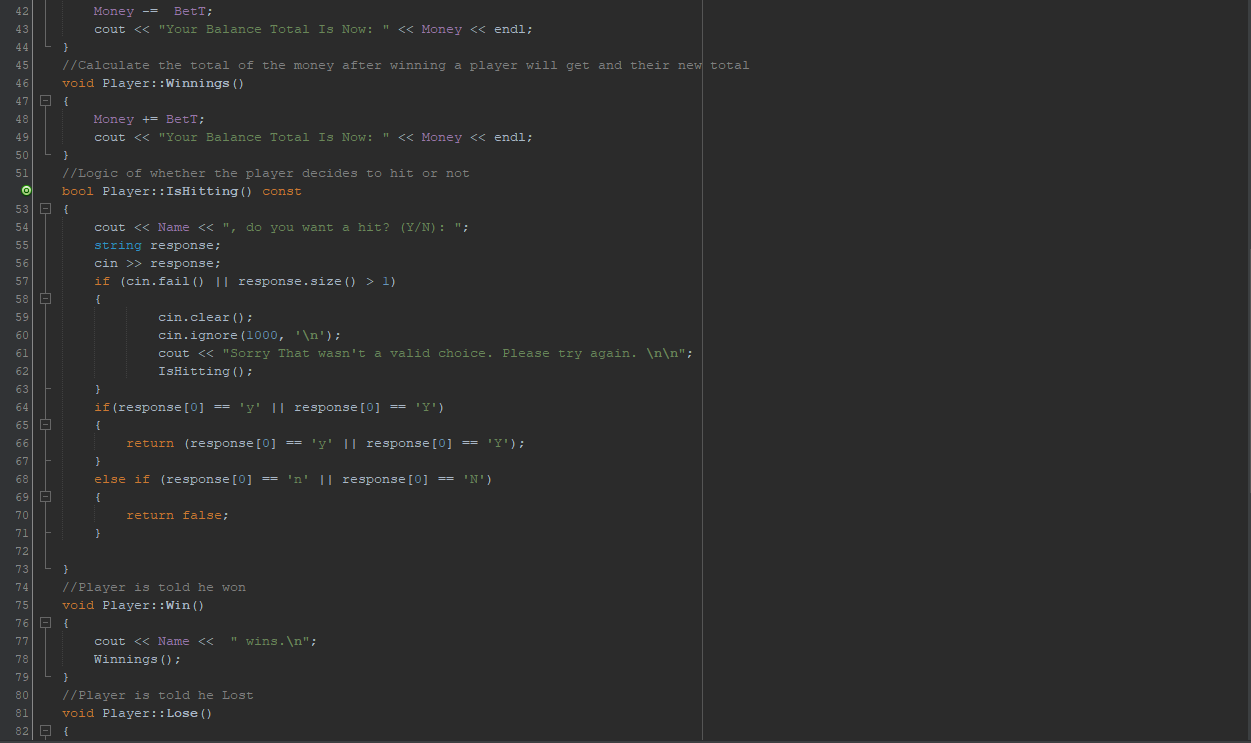


**Hand.h**

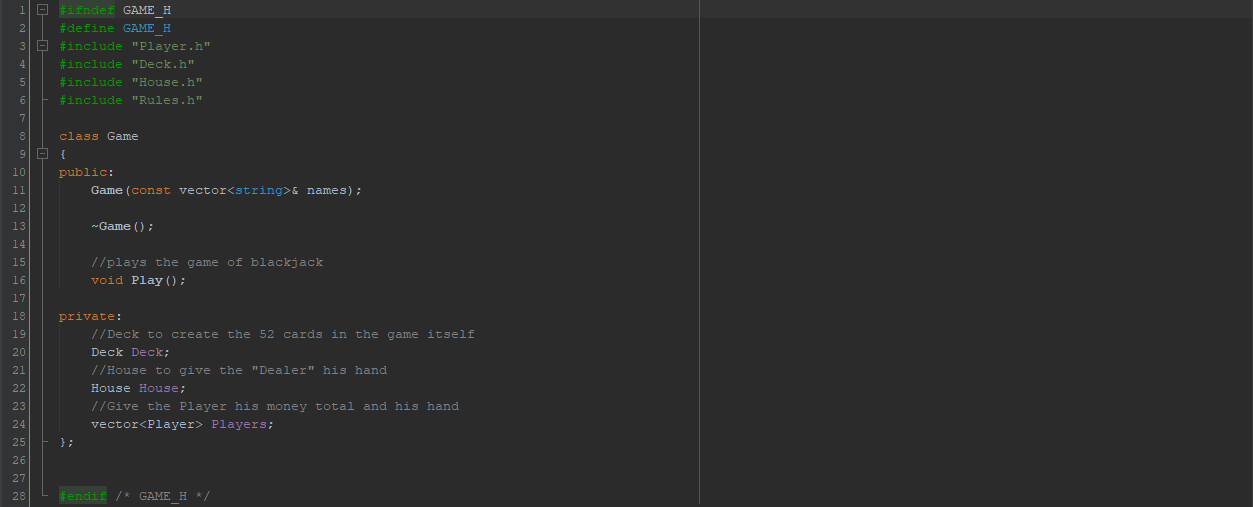
**House.cpp**

**House.h**

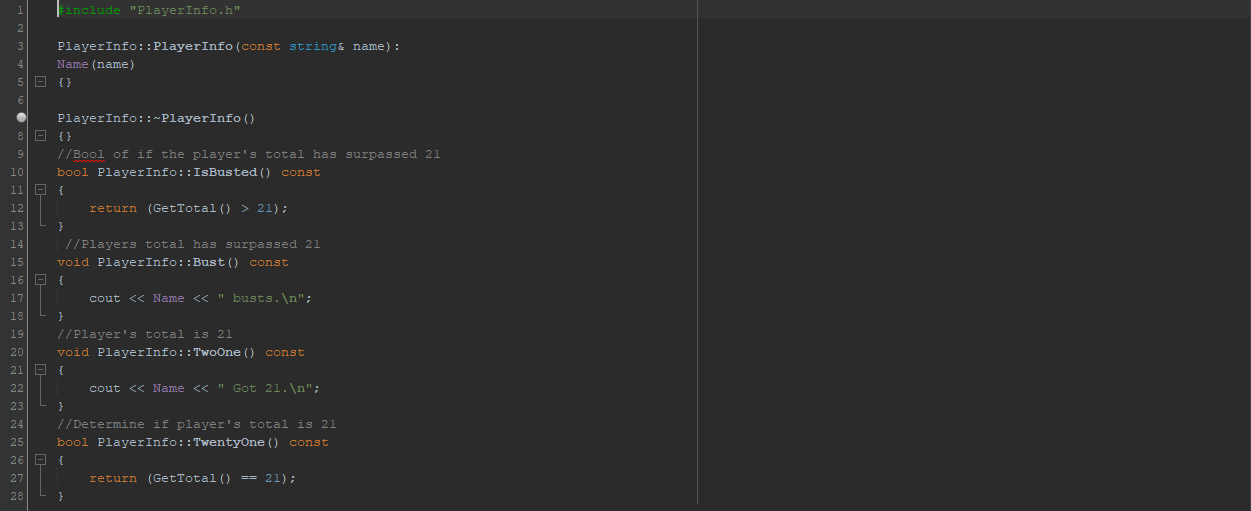
**Player.cpp**



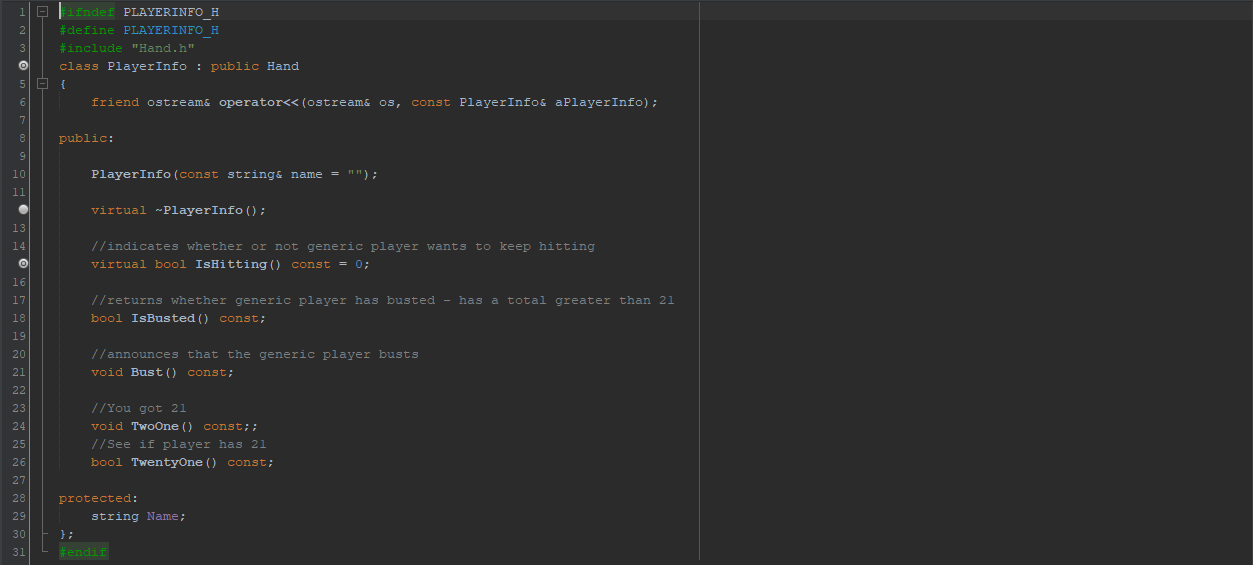
**Player.h**

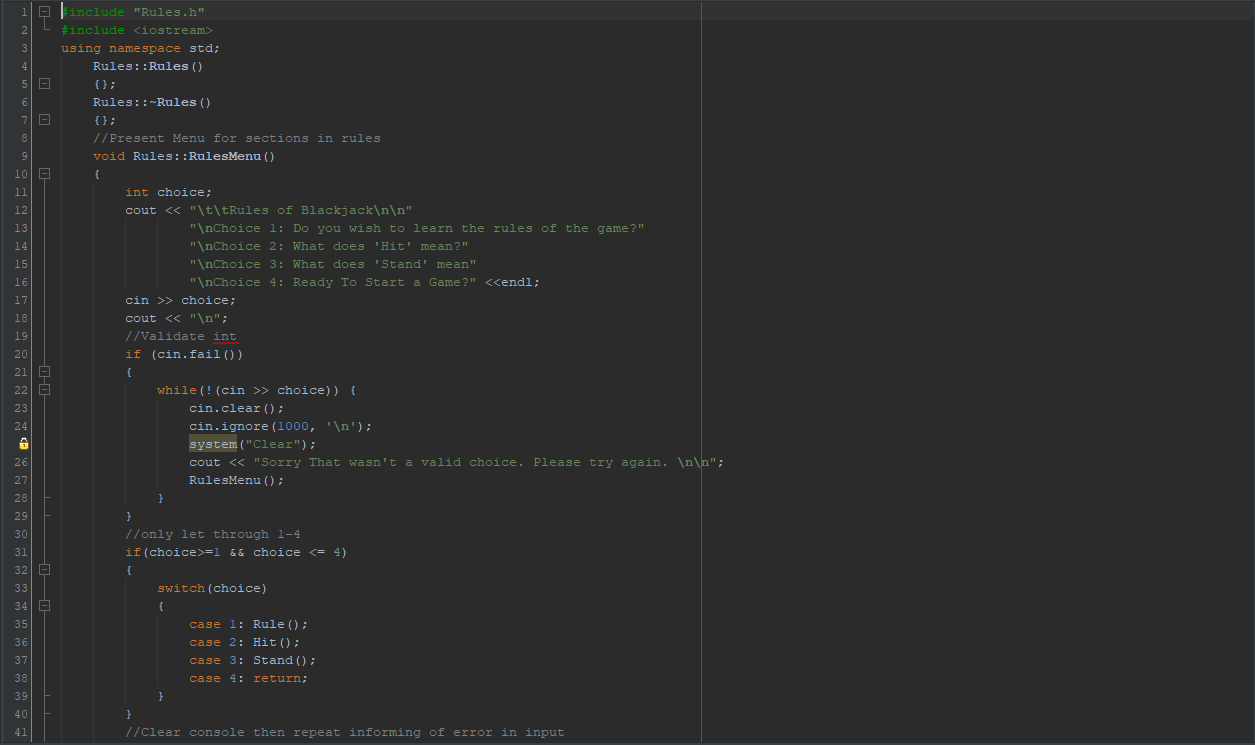


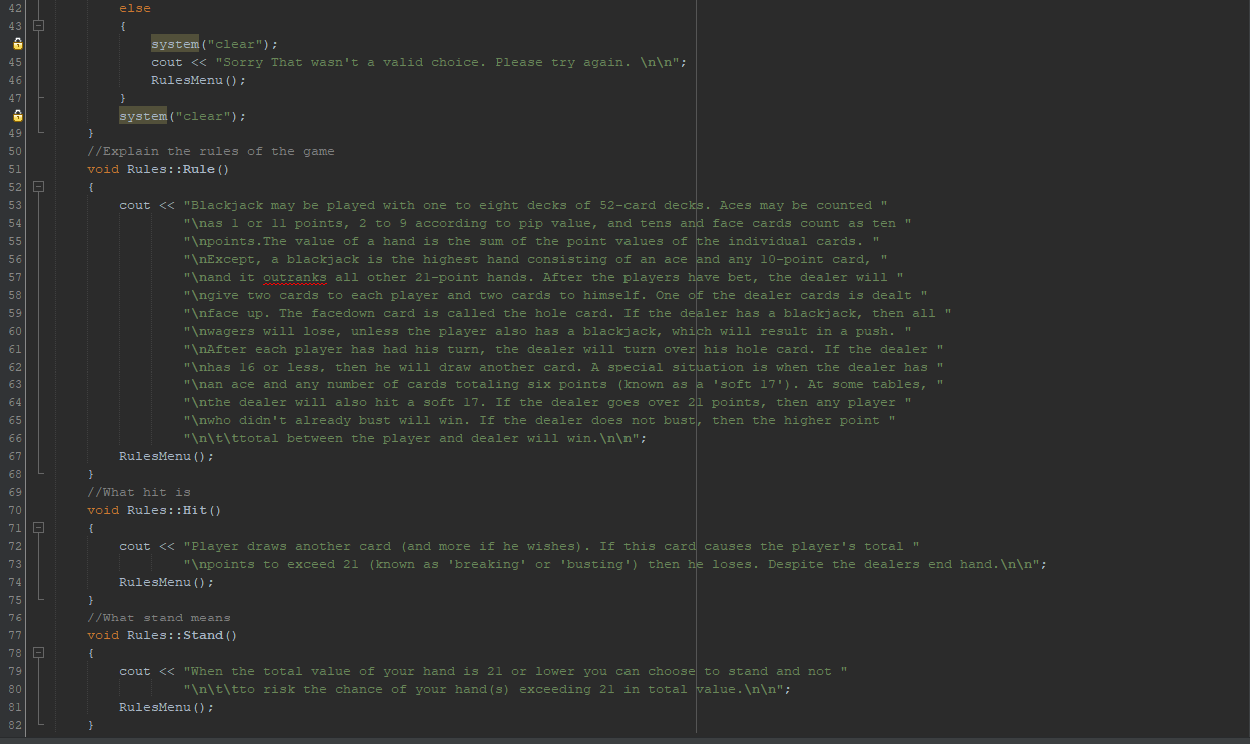
**PlayerInfo.cpp**



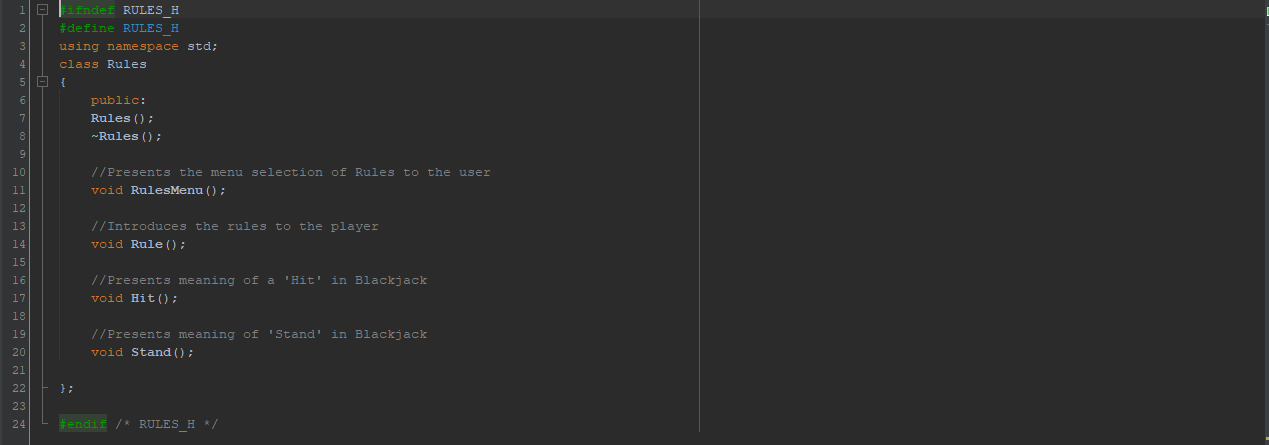
**PlayerInfo.h**

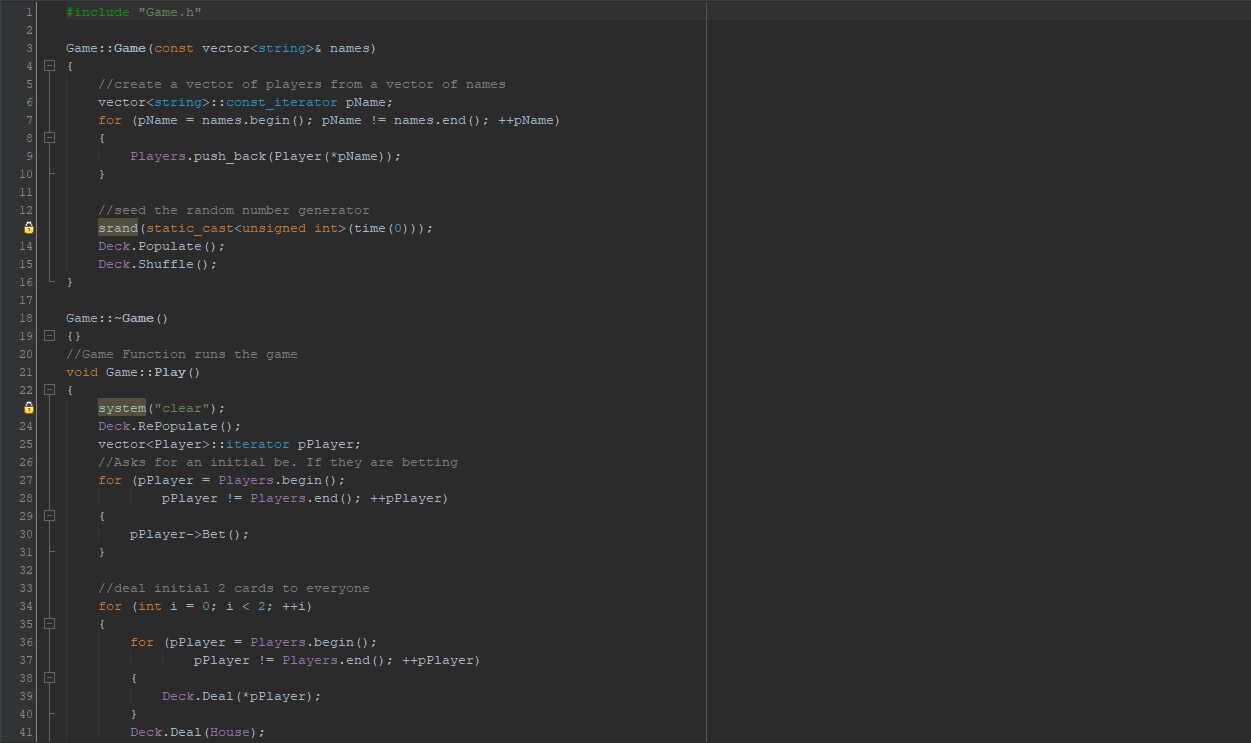
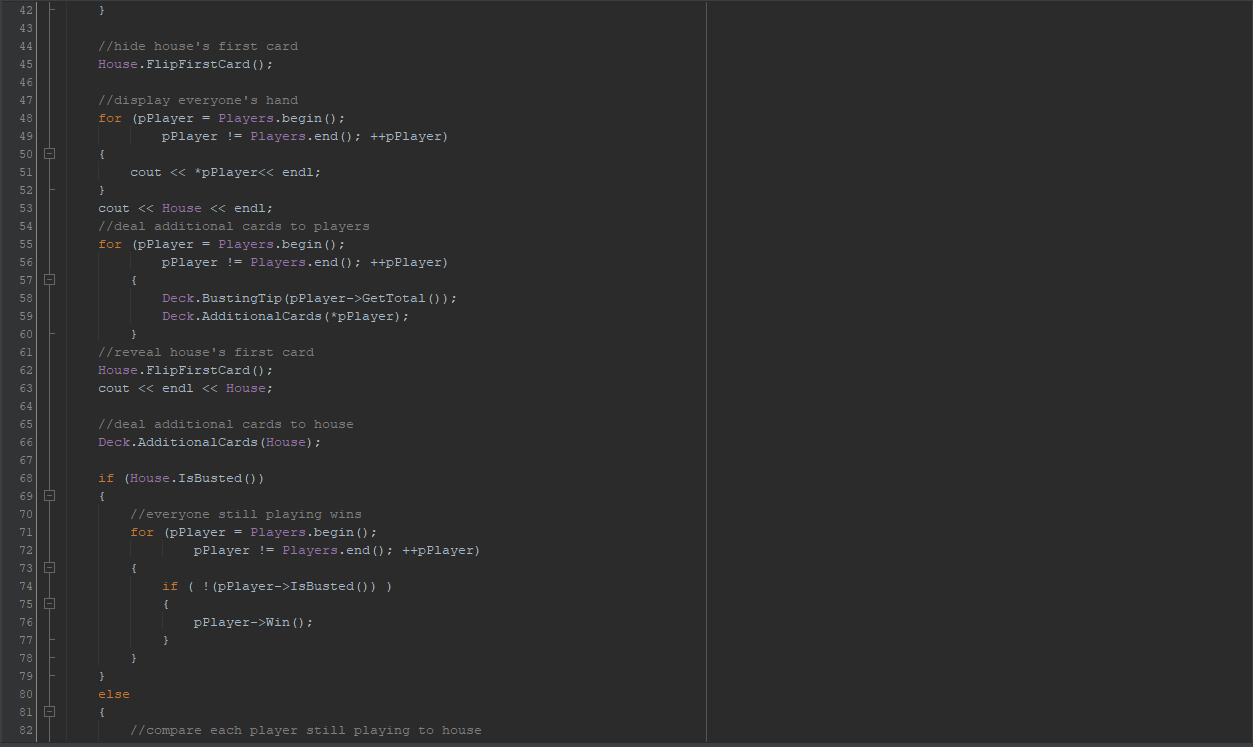
****

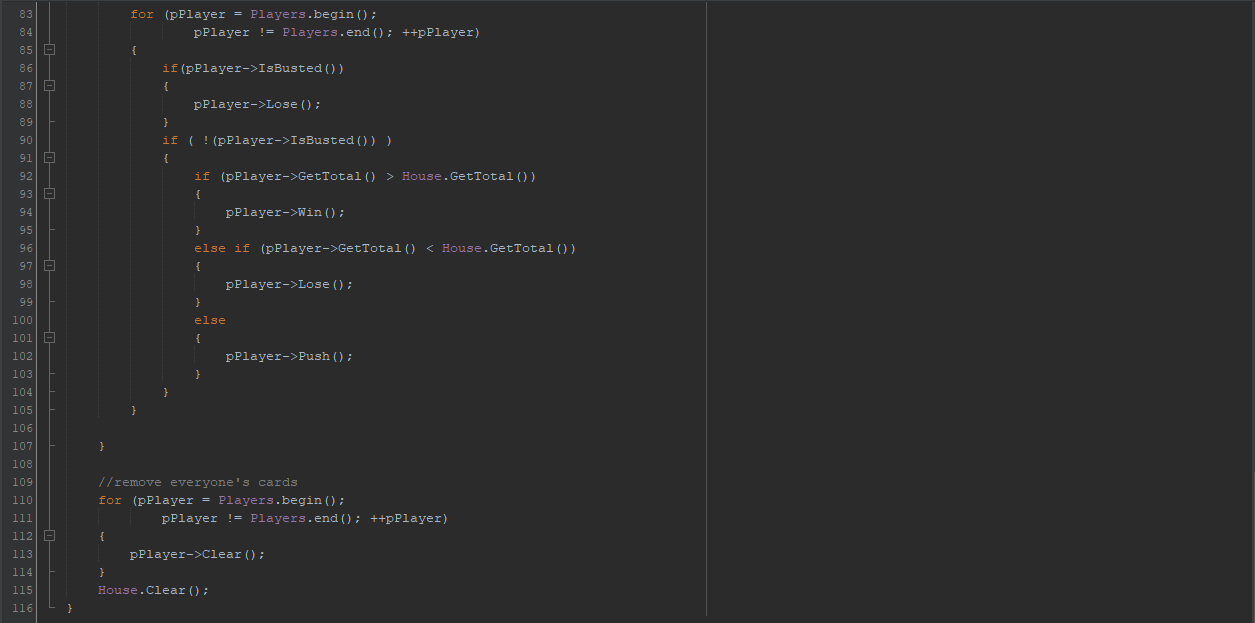
**Rules.cpp**

****

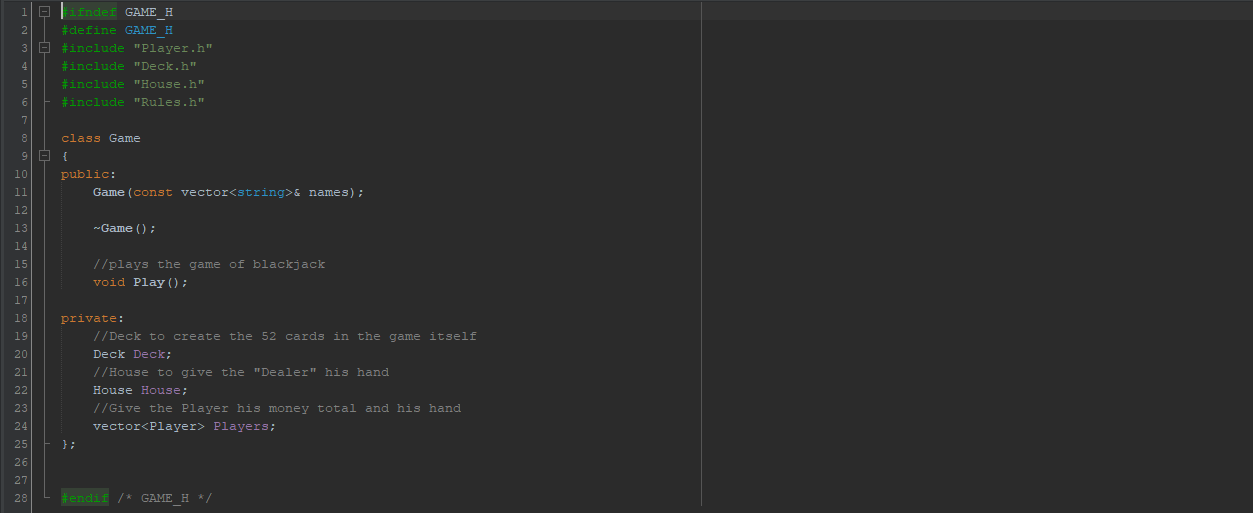
**Rules.h**

****

**Game.cpp**

****

**Game.h**

****