Project 1

<Blackjack>

Course:

CIS-17C

Section:

40369

Date:

April 26, 2020

Name:

Daniel Monges

Introduction:

Blackjack is the second most popular game played at casinos, only behind poker. Because of this I wanted to create a game that will allow the user to play and practice blackjack. Not only that but the program also allows the user to make bets, giving them an accurate portrayal of the game. I started working on the project on April 19, 2020 and I have spent about 3-5 hours each day, totalling to 21-35 hours in total on this project. The project has 714 lines of code and 126 blank lines. There are 3 different classes, there are class Player, class Card, and class Event. You can find the github post here through this link.

https://github.com/DanielMonges/Project-1

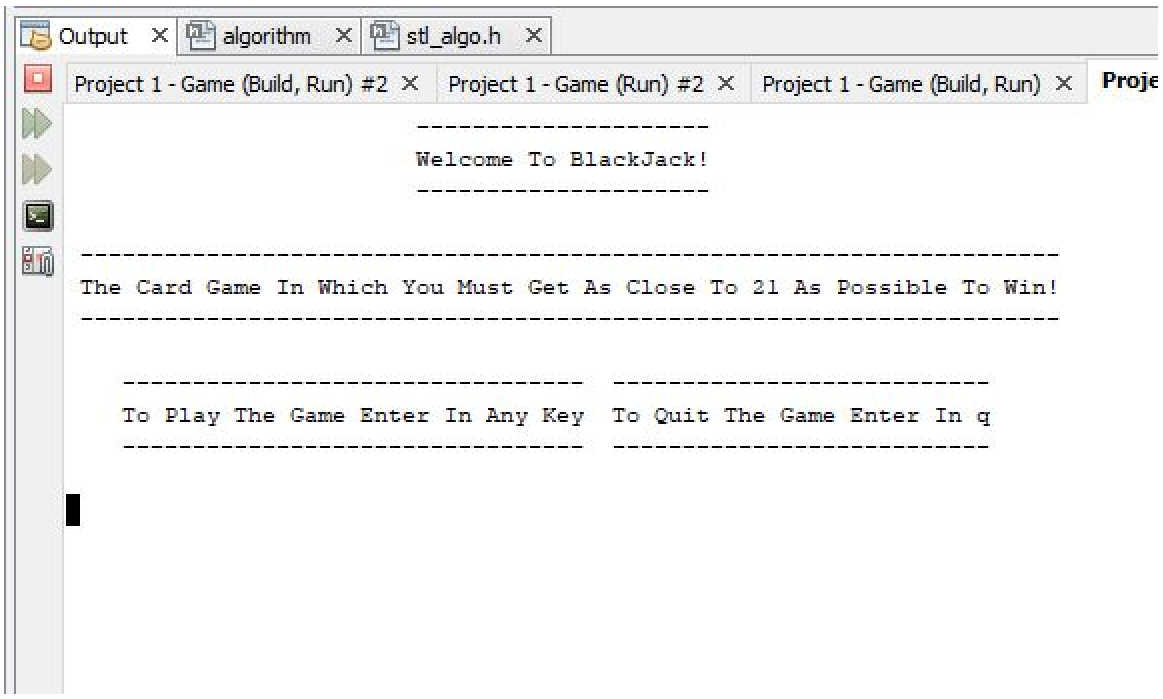
Game Rules:

The goal of the game is to have the hand that has a value of 21 or has the closest value to 21. Normally the game would be played with multiple decks however in this program it is played with one 52 standard deck. Aces have a value of 1 or 11 while cards 2-9 all have their designated number as their value. 10s, Jacks, Queens, and Kings all have a value of 10. Scoring over 21 in points will cause you to automatically lose regardless of what the other player has. A blackjack is the most powerful hand, as it consists of an Ace and any card that has a value of 10, having a total value of 21. Each player is given 2 cards before anything is done. There are 3 different choices the player can make after the cards have been dealt: Hit, Stay, or Double Down. Hitting is when you ask for another card, and you can keep doing this till you are satisfied or till you bust. Staying is when you stay with your current hand and don't take in any extra cards. Doing a double down is when you double your bet and you take only 1 extra card.

Description Of Code:

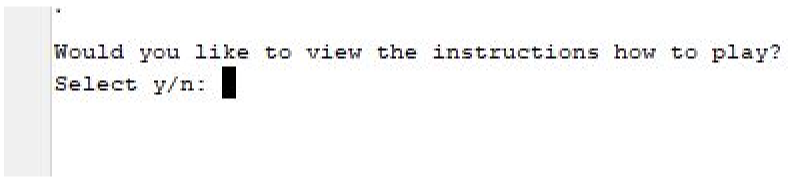
I have the main file which is called main.cpp and I have 3 other .h and .cpp files. There are Player.h, Card.h, and Event.h, not only that but there are also their respective .cpp files also. All the .h files have are the classes (which is named after the file name). The .cpp files have all the functions needed in order to make the class functions work (for each respective file.) Everything else is included inside of the main file.

Sample Input/Output:


Output
algorithm
stl_algo.h
Project 1 - Game (Build, Run) #2
Project 1 - Game (Run) #2
Project 1 - Game (Build, Run)
Proje

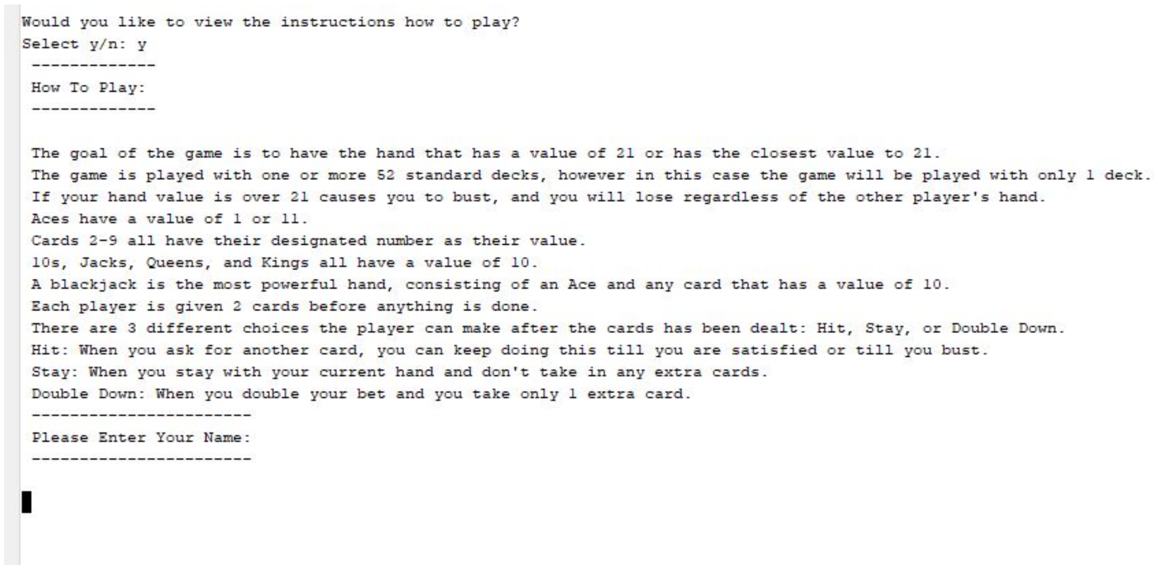
```
                         ---------------------
                         Welcome To BlackJack!
                         ---------------------


----------------------------------------------------------------------
The Card Game In Which You Must Get As Close To 21 As Possible To Win!
----------------------------------------------------------------------


   ---------------------------------  ---------------------------
   To Play The Game Enter In Any Key  To Quit The Game Enter In q
   ---------------------------------  ---------------------------
```

This is the screen that is shown the moment the program is loaded up. Pressing q will cause the game to end while pressing anything else will make the game continue.



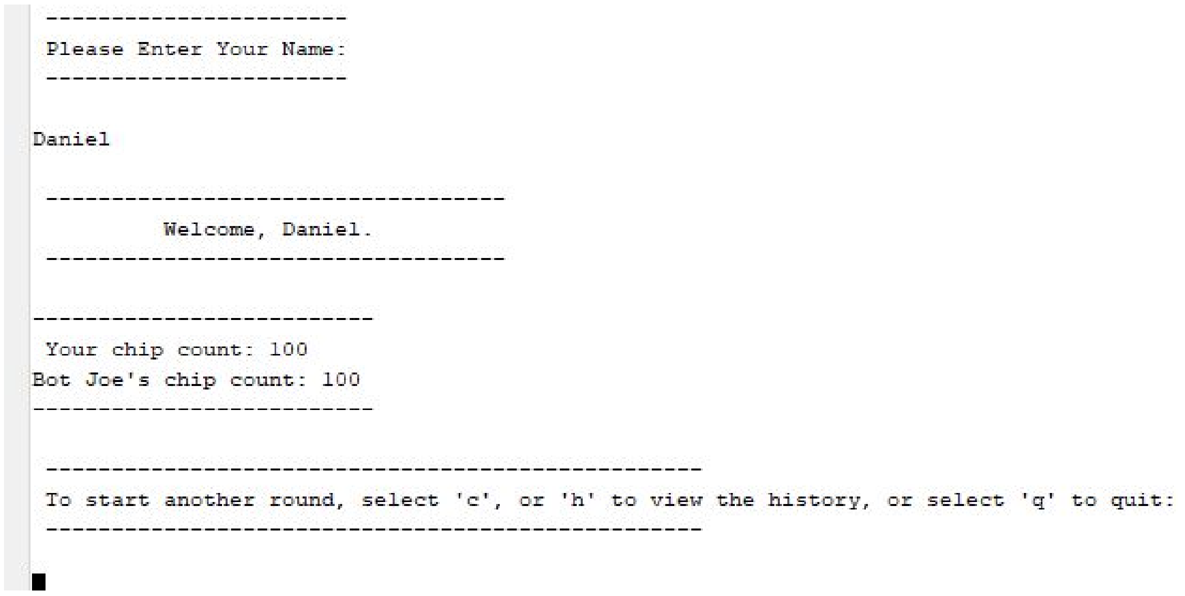
```
.
Would you like to view the instructions how to play?
Select y/n: █
```

The game will then ask the person if they want to see how to play the game.



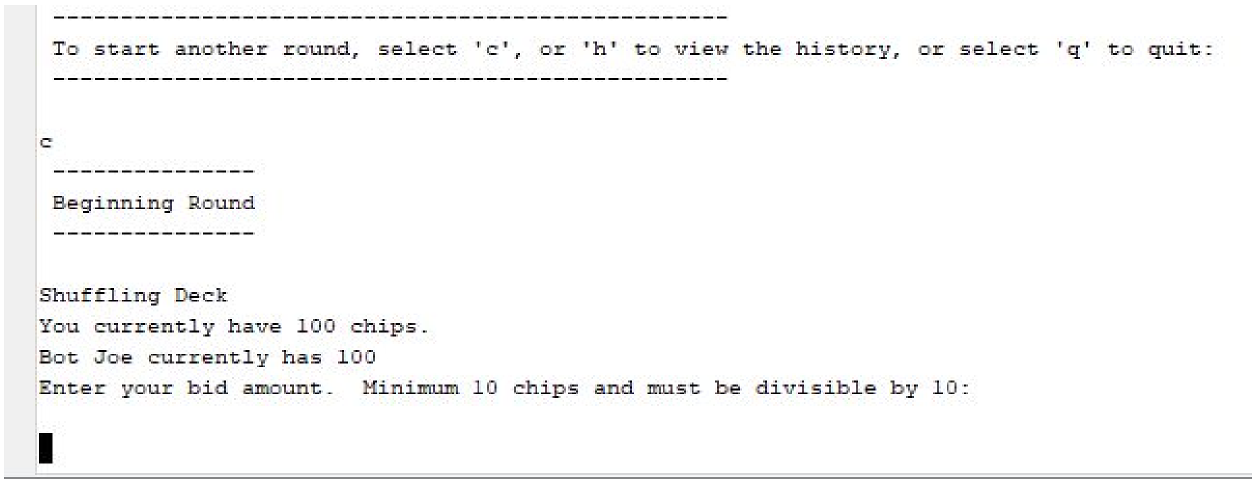
```
Would you like to view the instructions how to play?
Select y/n: y
 -------------
 How To Play:
 -------------

 The goal of the game is to have the hand that has a value of 21 or has the closest value to 21.
 The game is played with one or more 52 standard decks, however in this case the game will be played with only 1 deck.
 If your hand value is over 21 causes you to bust, and you will lose regardless of the other player's hand.
 Aces have a value of 1 or 11.
 Cards 2-9 all have their designated number as their value.
 10s, Jacks, Queens, and Kings all have a value of 10.
 A blackjack is the most powerful hand, consisting of an Ace and any card that has a value of 10.
 Each player is given 2 cards before anything is done.
 There are 3 different choices the player can make after the cards has been dealt: Hit, Stay, or Double Down.
 Hit: When you ask for another card, you can keep doing this till you are satisfied or till you bust.
 Stay: When you stay with your current hand and don't take in any extra cards.
 Double Down: When you double your bet and you take only 1 extra card.
 -----------------------
 Please Enter Your Name:
 -----------------------

█
```

This screen is shown when the player inputs y, if they inputted n then it would ask them to enter their name right away.



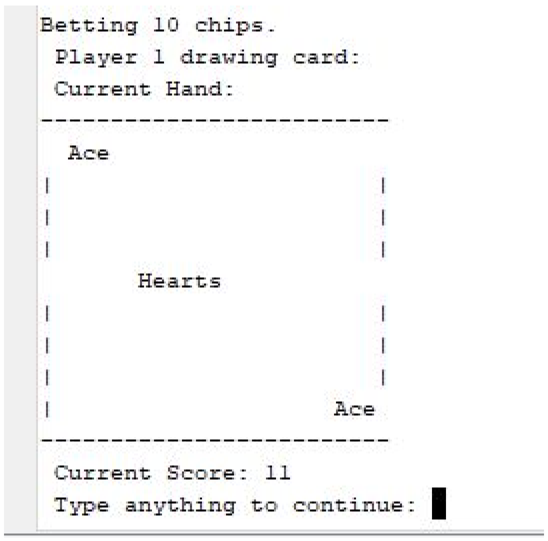
```
 -----------------------
 Please Enter Your Name:
 -----------------------

Daniel

 -----------------------------------
         Welcome, Daniel.
 -----------------------------------

--------------------------
 Your chip count: 100
Bot Joe's chip count: 100
--------------------------

 --------------------------------------------------
 To start another round, select 'c', or 'h' to view the history, or select 'q' to quit:
 --------------------------------------------------
```

You are given a welcome sign and are told of the total amount of chips owned by both you and the enemy, which in this program is Bot Joe. It also asks you if you want to start a round, view past games, or to quit.



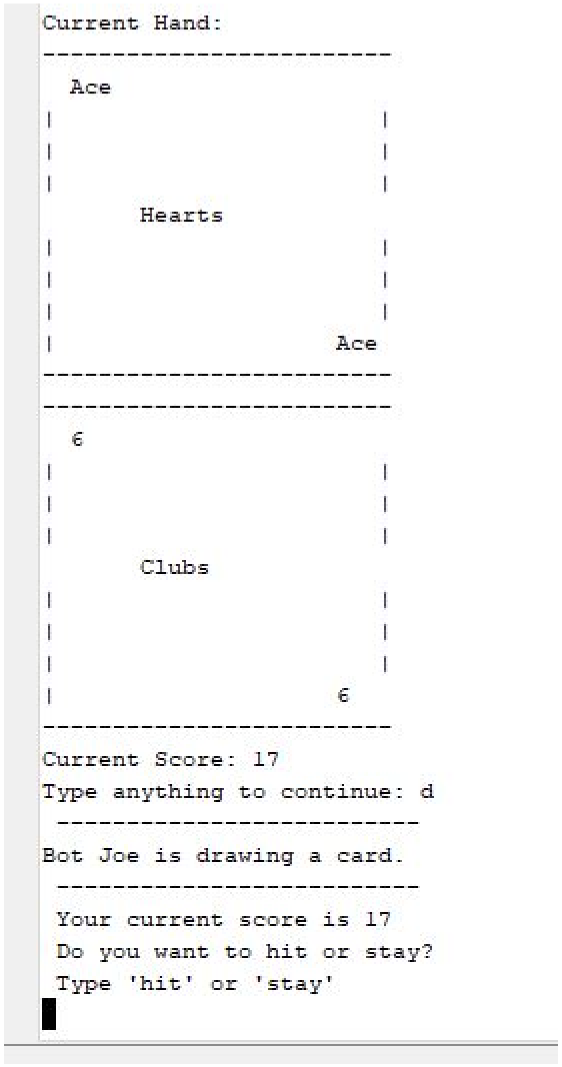
```
 --------------------------------------------------
 To start another round, select 'c', or 'h' to view the history, or select 'q' to quit:
 --------------------------------------------------

c
 ---------------
 Beginning Round
 ---------------

Shuffling Deck
You currently have 100 chips.
Bot Joe currently has 100
Enter your bid amount.  Minimum 10 chips and must be divisible by 10:
```

You are asked to input the amount of chips that you want to bet, the minimum is 10 and it has to be divisible by 10 also.



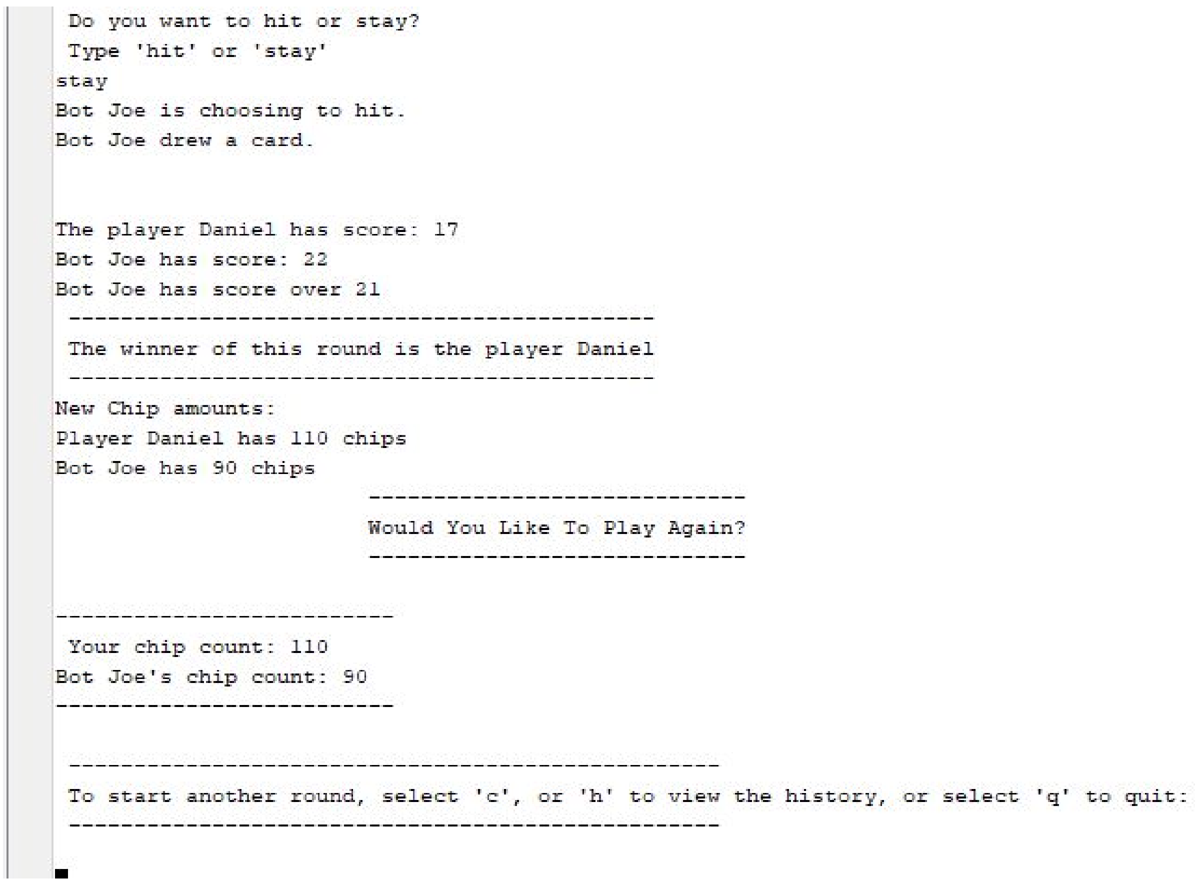
```
Betting 10 chips.
 Player 1 drawing card:
 Current Hand:
-------------------------
  Ace
|                        |
|                        |
|                        |
       Hearts
|                        |
|                        |
|                        |
|                   Ace
-------------------------
 Current Score: 11
 Type anything to continue:
```

You are given your first card and the only thing you can do is continue as the cards are being passed out.



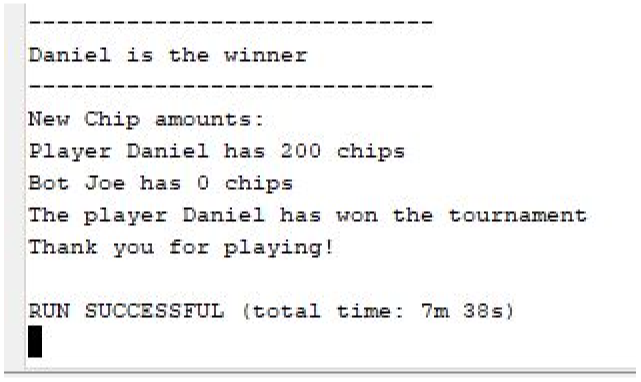
```
Current Hand:
-------------------------
  Ace
|                       |
|                       |
|                       |
       Hearts
|                       |
|                       |
|                       |
|                    Ace
-------------------------
-------------------------
  6
|                       |
|                       |
|                       |
       Clubs
|                       |
|                       |
|                       |
|                    6
-------------------------
Current Score: 17
Type anything to continue: d
 --------------------------
Bot Joe is drawing a card.
 --------------------------
 Your current score is 17
 Do you want to hit or stay?
 Type 'hit' or 'stay'
```

You are given your two cards and are given your total score. You are given the option to hit or to stay. Hitting will give you another card while staying will continue the game.



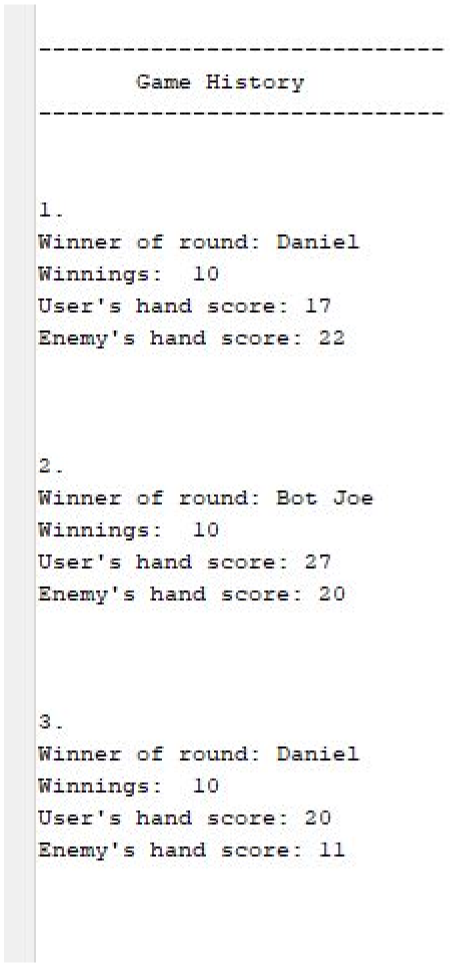
```
 Do you want to hit or stay?
 Type 'hit' or 'stay'
stay
Bot Joe is choosing to hit.
Bot Joe drew a card.


The player Daniel has score: 17
Bot Joe has score: 22
Bot Joe has score over 21
 ---------------------------------------------
 The winner of this round is the player Daniel
 ---------------------------------------------
New Chip amounts:
Player Daniel has 110 chips
Bot Joe has 90 chips
                        -----------------------------
                        Would You Like To Play Again?
                        -----------------------------


--------------------------
 Your chip count: 110
Bot Joe's chip count: 90
--------------------------

 --------------------------------------------------
 To start another round, select 'c', or 'h' to view the history, or select 'q' to quit:
 --------------------------------------------------
```

The game tells you your score and your enemies score. After that it determines who is the winner and who gains the chips. It then asks if you would like to play again, causing the game to loop until someone loses all their chips.



```
-----------------------------
Daniel is the winner
-----------------------------
New Chip amounts:
Player Daniel has 200 chips
Bot Joe has 0 chips
The player Daniel has won the tournament
Thank you for playing!

RUN SUCCESSFUL (total time: 7m 38s)
```

After a player loses all their chips then the winner is chosen and the program ends.



```
-----------------------------
       Game History
-----------------------------


1.
Winner of round: Daniel
Winnings:  10
User's hand score: 17
Enemy's hand score: 22



2.
Winner of round: Bot Joe
Winnings:  10
User's hand score: 27
Enemy's hand score: 20



3.
Winner of round: Daniel
Winnings:  10
User's hand score: 20
Enemy's hand score: 11
```

This is what the history page looks like. Bare in mind that it will be different for everyone as it takes in only the games that were just played.

CheckList:

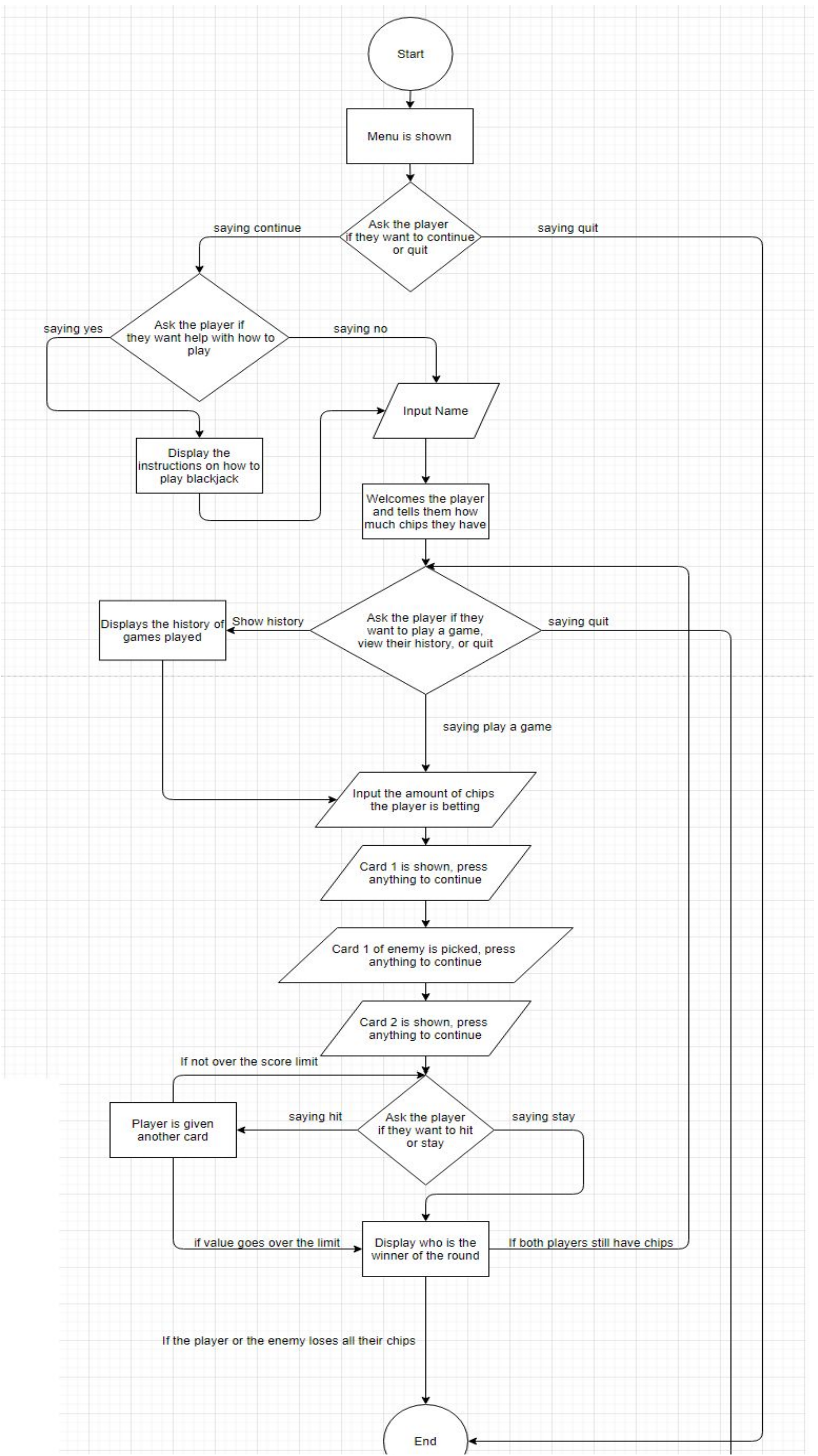
Type	**Variable Name**	**Description**
int	currentHandValue	The value of the current hand that the player has
	currentAmountOfCards	The amount of cards that the player currently has
	totalChipCount	The total amount of chips that the player has
	value	The value assigned to the cards
	roundWinnings	The amount of chips that the player won
	userScore	The score that the user has
	enemyScore	The score given to the enemy player
	cardValue[13]	The array holding the value of the cards

	Count	Use to count how many cards were given out to the players
	bet	The place where the player puts the amount they want to bet
	randomCard	Picks a random card to be given
	ans	Answer that the player gave when asked questions
String	roundWinner	Says who is the winner of the round
	suit	Stores the name of the 4 different card suits
	suitNumber	Stores the number that each card is
	name	Name of the player
	suits[4]	Array storing the information about the 4 different suits
	suitNumber[13]	Stores the values of the different cards found inside of the suits
List	<Event> gameHistory	Stores the past games that the player played in order

		for them to see
Stack	<Card*> cardStack	Used to stack the data of the cards that were already given out
forward iterator		Forward iterators were used in the program to allow input and outputs
Copy	(cardArray, cardArray + 52, randomizedCards)	Used to copy the values for cardArray

Documentation Of Code:

Flowchart:

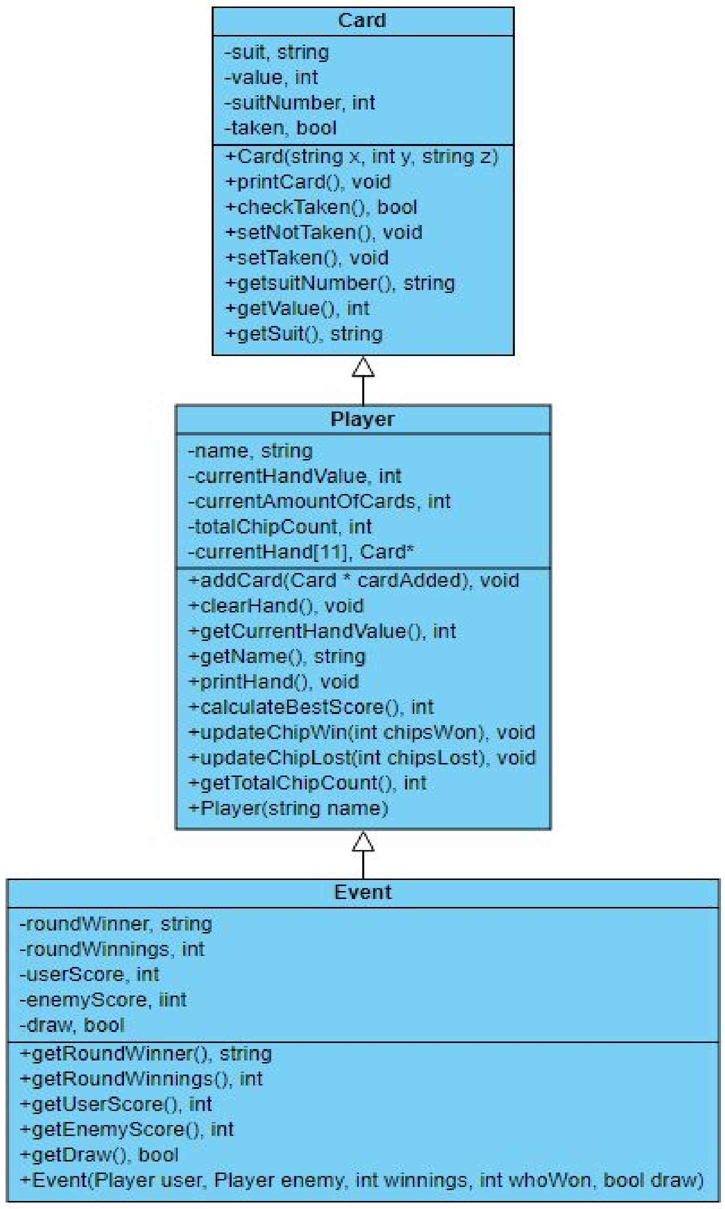

Start
Menu is shown
Ask the player if they want to continue or quit
saying continue
saying quit
Ask the player if they want help with how to play
saying yes
saying no
Input Name
Display the instructions on how to play blackjack
Welcomes the player and tells them how much chips they have
Ask the player if they want to play a game, view their history, or quit
Show history
saying quit
Displays the history of games played
saying play a game
Input the amount of chips the player is betting
Card 1 is shown, press anything to continue
Card 1 of enemy is picked, press anything to continue
Card 2 is shown, press anything to continue
If not over the score limit
Ask the player if they want to hit or stay
saying hit
saying stay
Player is given another card
if value goes over the limit
Display who is the winner of the round
If both players still have chips
If the player or the enemy loses all their chips
End

Pseudo-Code:

This is the pseudo-code that I made when I first started the project. It is much less than I ended up with as I came up with newer stuff to add later on.

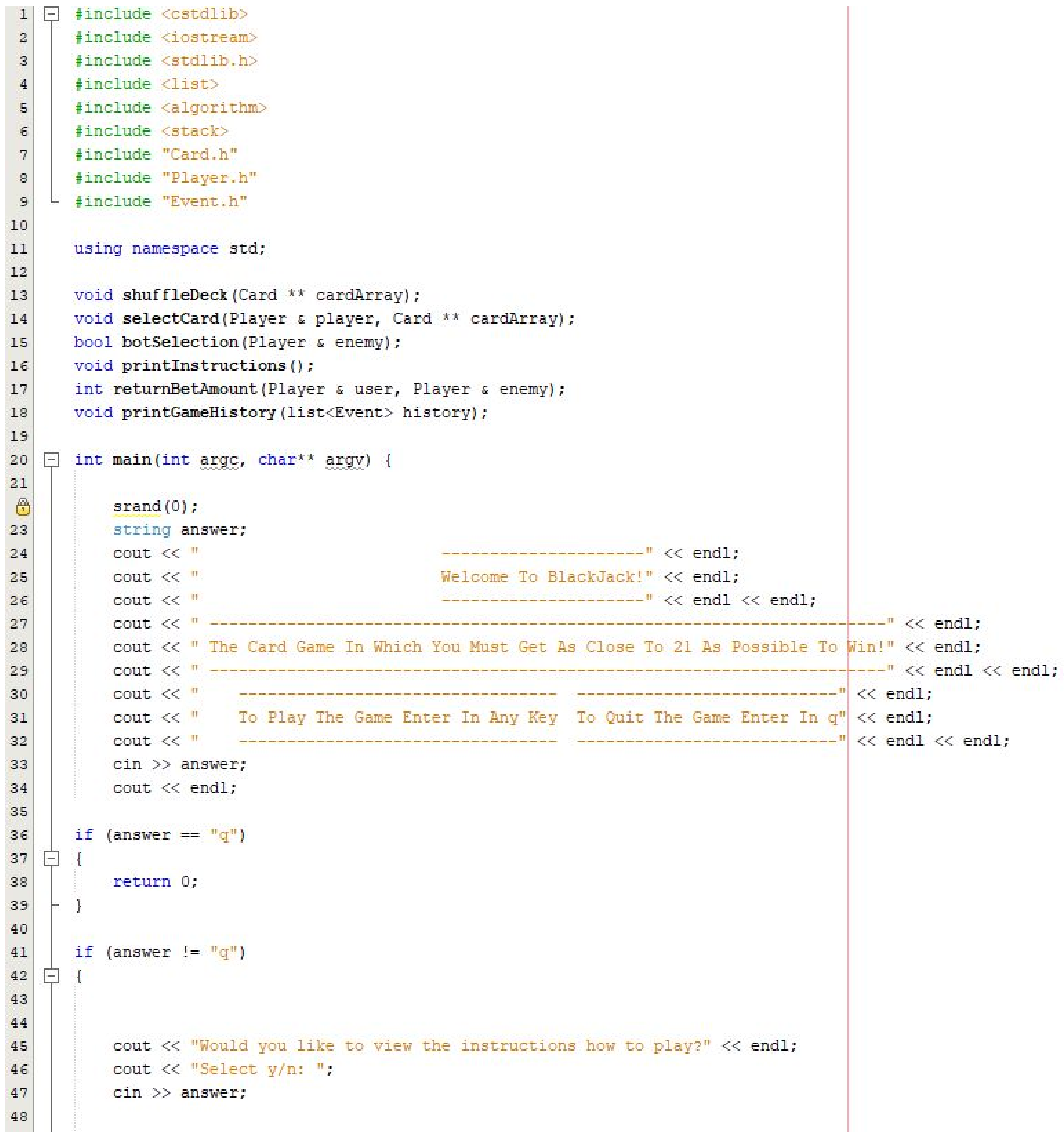
```
1.  Welcome the player
2.  Ask if they want to learn the instructions on how to play
3.  (if they said yes, then show the rules, if they said no, then continue into the game)
4.  Ask the player to make a bet, lowest is 10
5.  1 card is given to player 1
6.  1 card is given to enemy player
7.  1 card given to player 1
8.  1 card given to enemy player
9.  Look at the cards given
10. (both cards will be displayed at this point)
11. Choose hit or stay
12. (show all the cards if they choose hit, continue if they hit stay)
13. You either win or lose
```

UML Class Diagram:

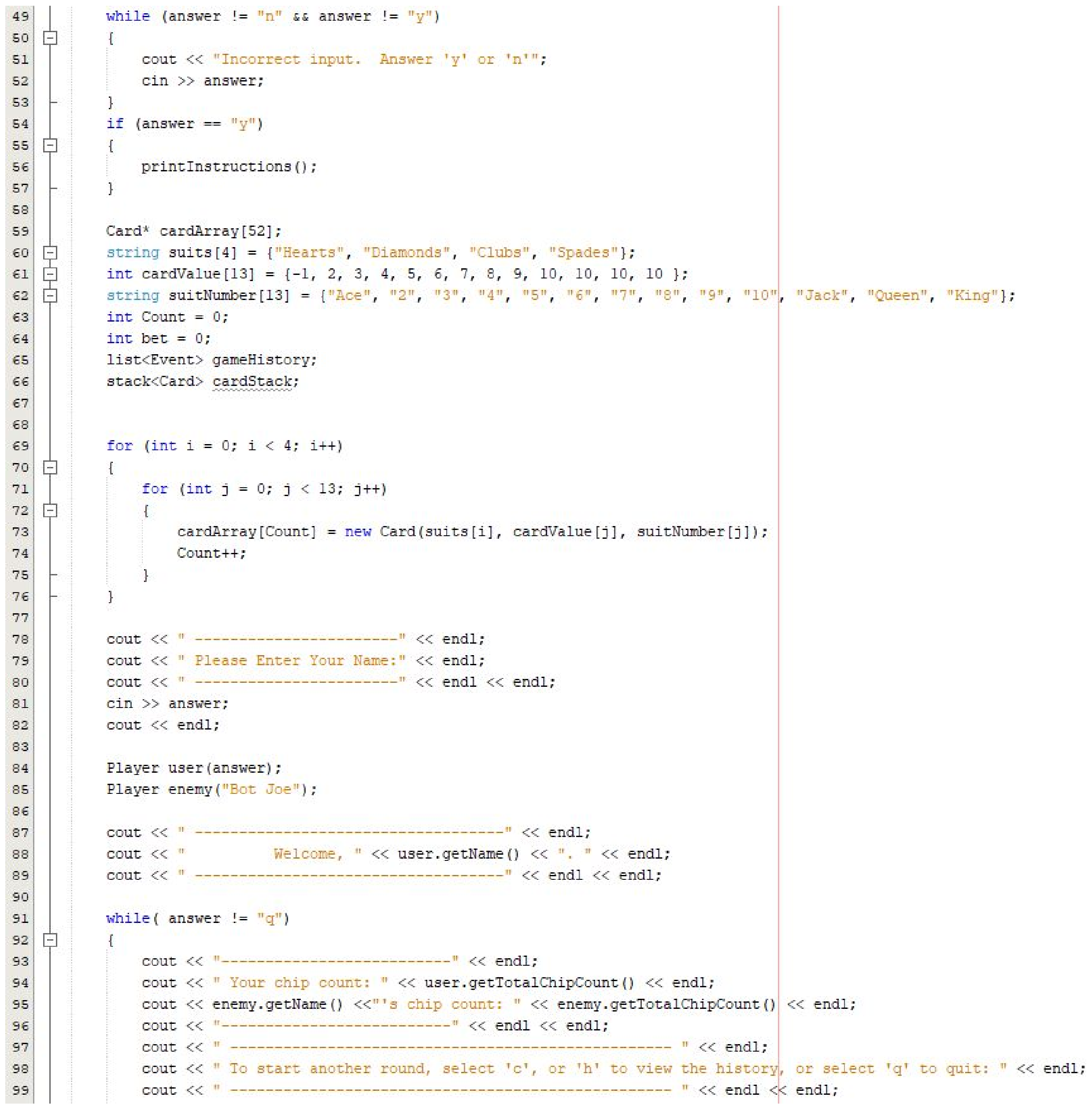


Program:

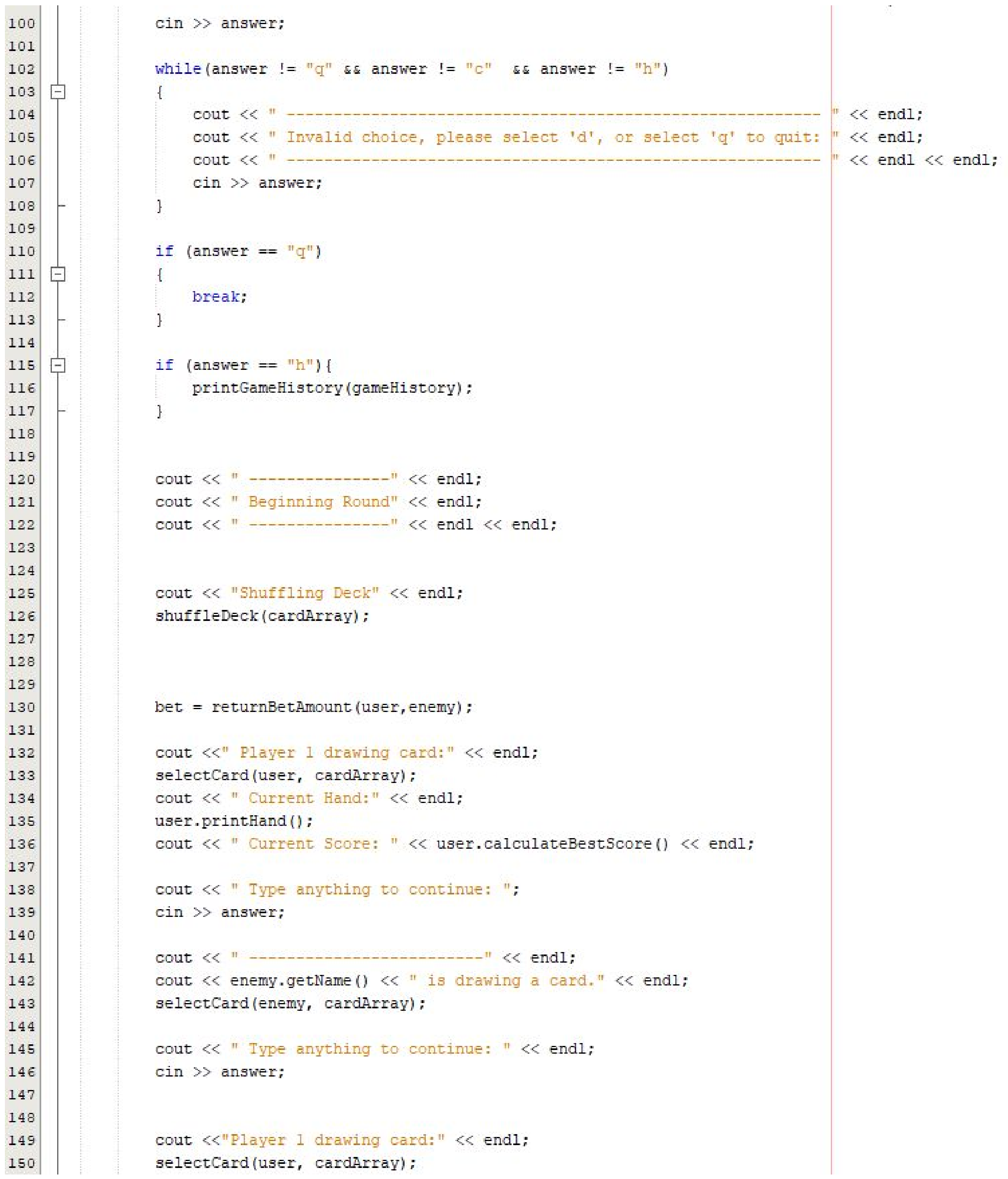
Main.cpp



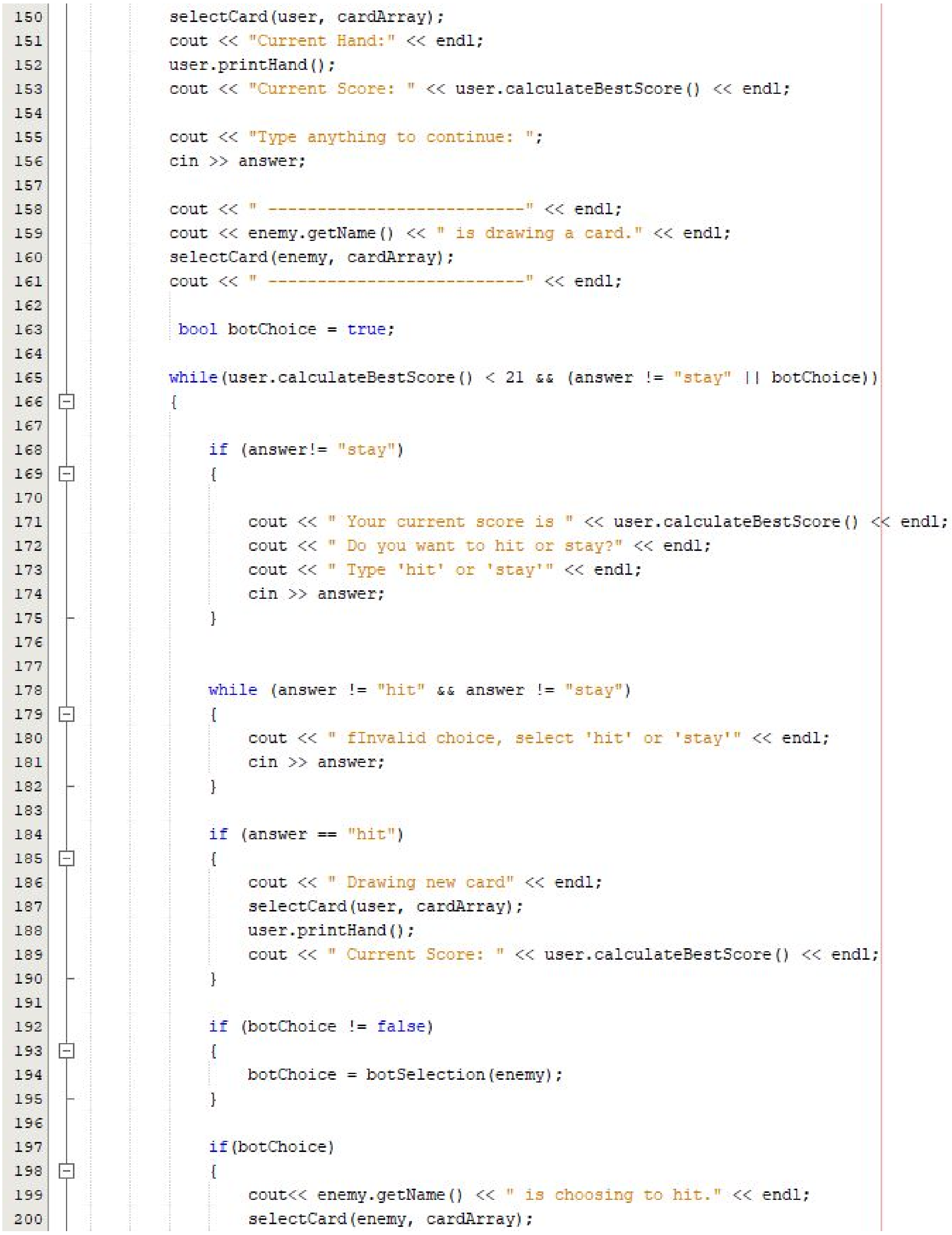
```
#include <cstdlib>
#include <iostream>
#include <stdlib.h>
#include <list>
#include <algorithm>
#include <stack>
#include "Card.h"
#include "Player.h"
#include "Event.h"

using namespace std;

void shuffleDeck(Card ** cardArray);
void selectCard(Player & player, Card ** cardArray);
bool botSelection(Player & enemy);
void printInstructions();
int returnBetAmount(Player & user, Player & enemy);
void printGameHistory(list<Event> history);

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

    srand(0);
    string answer;
    cout << "                        ---------------------" << endl;
    cout << "                        Welcome To BlackJack!" << endl;
    cout << "                        ---------------------" << endl << endl;
    cout << " ----------------------------------------------------------------------" << endl;
    cout << " The Card Game In Which You Must Get As Close To 21 As Possible To Win!" << endl;
    cout << " ----------------------------------------------------------------------" << endl << endl;
    cout << "    ---------------------------------  ---------------------------" << endl;
    cout << "    To Play The Game Enter In Any Key  To Quit The Game Enter In q" << endl;
    cout << "    ---------------------------------  ---------------------------" << endl << endl;
    cin >> answer;
    cout << endl;

if (answer == "q")
{
    return 0;
}

if (answer != "q")
{


    cout << "Would you like to view the instructions how to play?" << endl;
    cout << "Select y/n: ";
    cin >> answer;

```



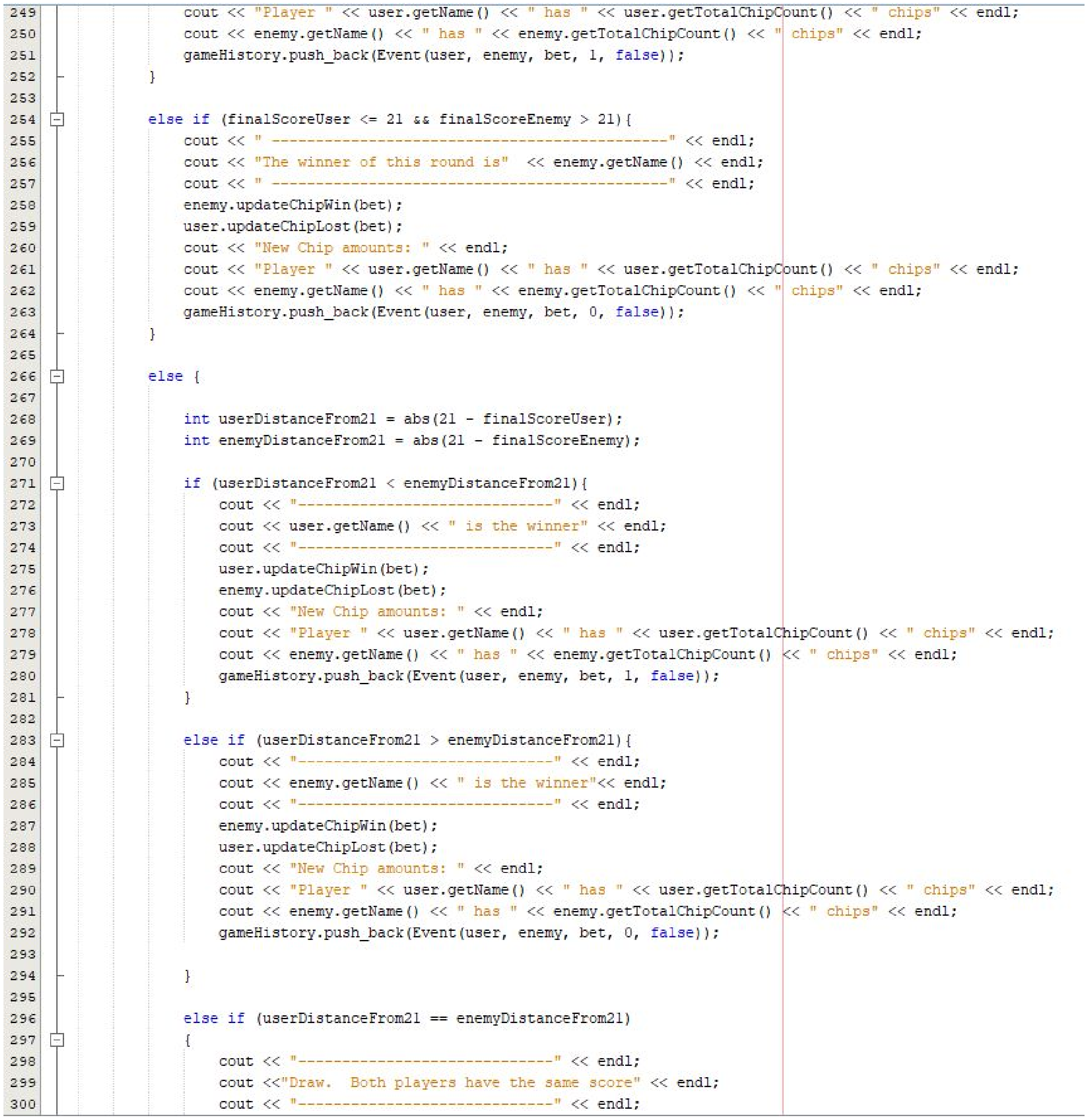
```cpp
    while (answer != "n" && answer != "y")
    {
        cout << "Incorrect input.  Answer 'y' or 'n'";
        cin >> answer;
    }
    if (answer == "y")
    {
        printInstructions();
    }

    Card* cardArray[52];
    string suits[4] = {"Hearts", "Diamonds", "Clubs", "Spades"};
    int cardValue[13] = {-1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10 };
    string suitNumber[13] = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King"};
    int Count = 0;
    int bet = 0;
    list<Event> gameHistory;
    stack<Card> cardStack;


    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 13; j++)
        {
            cardArray[Count] = new Card(suits[i], cardValue[j], suitNumber[j]);
            Count++;
        }
    }

    cout << " -----------------------" << endl;
    cout << " Please Enter Your Name:" << endl;
    cout << " -----------------------" << endl << endl;
    cin >> answer;
    cout << endl;

    Player user(answer);
    Player enemy("Bot Joe");

    cout << " -----------------------------------" << endl;
    cout << "          Welcome, " << user.getName() << ". " << endl;
    cout << " -----------------------------------" << endl << endl;

    while( answer != "q")
    {
        cout << "--------------------------" << endl;
        cout << " Your chip count: " << user.getTotalChipCount() << endl;
        cout << enemy.getName() <<"'s chip count: " << enemy.getTotalChipCount() << endl;
        cout << "--------------------------" << endl << endl;
        cout << " -------------------------------------------------- " << endl;
        cout << " To start another round, select 'c', or 'h' to view the history, or select 'q' to quit: " << endl;
        cout << " -------------------------------------------------- " << endl << endl;
```



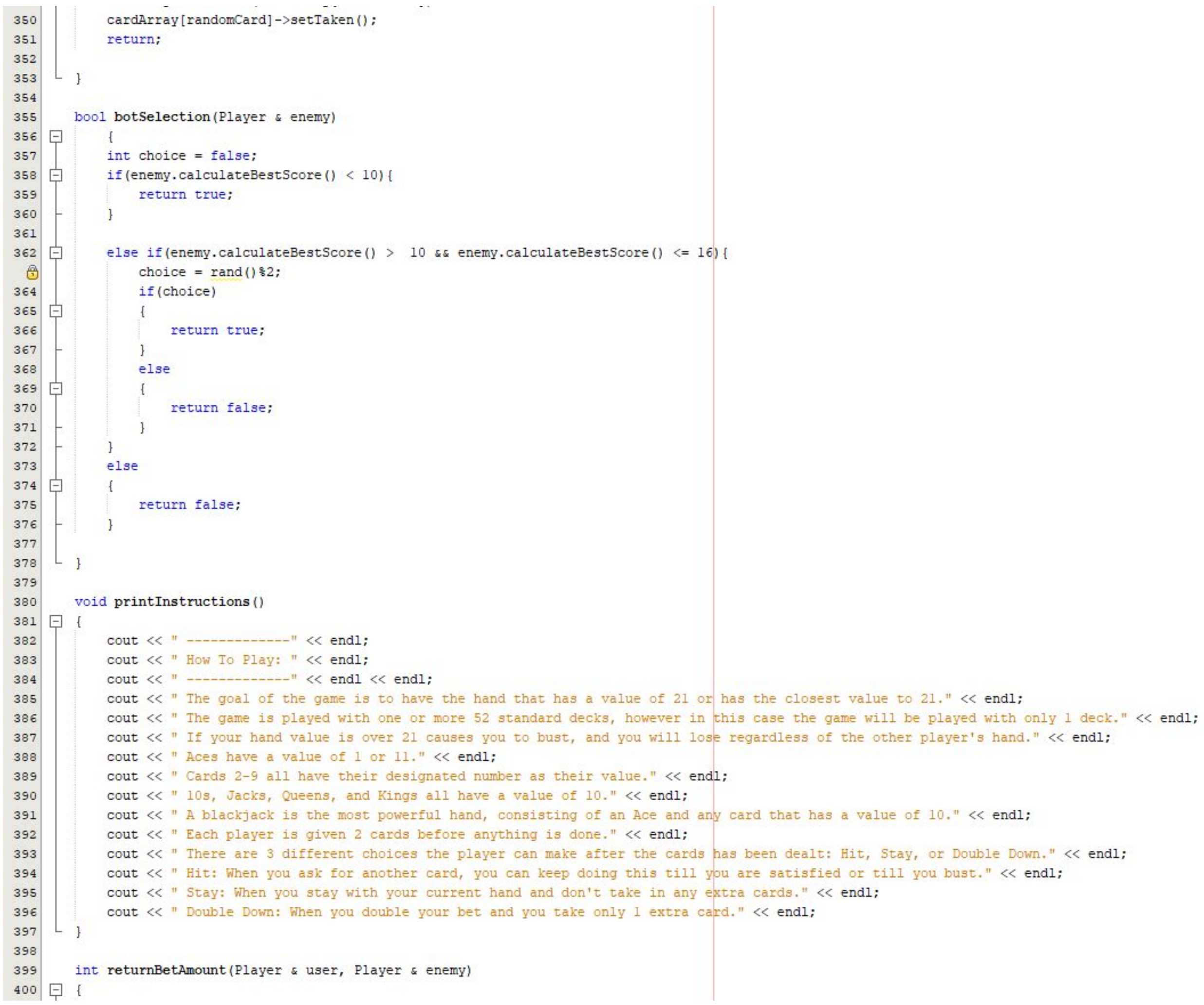
```cpp
            cin >> answer;

            while(answer != "q" && answer != "c"  && answer != "h")
            {
                cout << " --------------------------------------------------------- " << endl;
                cout << " Invalid choice, please select 'd', or select 'q' to quit: " << endl;
                cout << " --------------------------------------------------------- " << endl << endl;
                cin >> answer;
            }

            if (answer == "q")
            {
                break;
            }

            if (answer == "h"){
                printGameHistory(gameHistory);
            }


            cout << " ---------------" << endl;
            cout << " Beginning Round" << endl;
            cout << " ---------------" << endl << endl;


            cout << "Shuffling Deck" << endl;
            shuffleDeck(cardArray);



            bet = returnBetAmount(user,enemy);

            cout <<" Player 1 drawing card:" << endl;
            selectCard(user, cardArray);
            cout << " Current Hand:" << endl;
            user.printHand();
            cout << " Current Score: " << user.calculateBestScore() << endl;

            cout << " Type anything to continue: ";
            cin >> answer;

            cout << " -------------------------" << endl;
            cout << enemy.getName() << " is drawing a card." << endl;
            selectCard(enemy, cardArray);

            cout << " Type anything to continue: " << endl;
            cin >> answer;


            cout <<"Player 1 drawing card:" << endl;
            selectCard(user, cardArray);
```



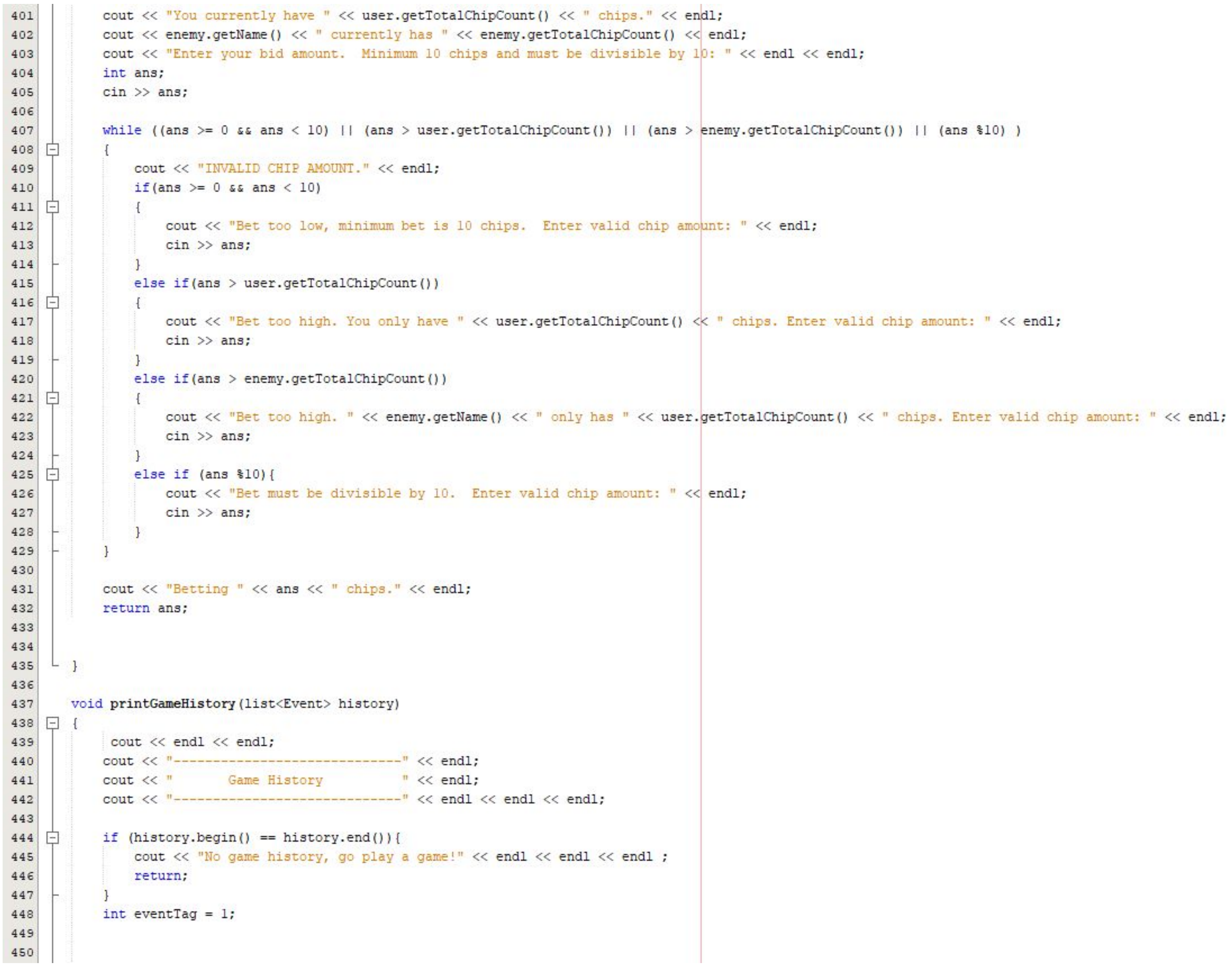
```
        selectCard(user, cardArray);
        cout << "Current Hand:" << endl;
        user.printHand();
        cout << "Current Score: " << user.calculateBestScore() << endl;

        cout << "Type anything to continue: ";
        cin >> answer;

        cout << " --------------------------" << endl;
        cout << enemy.getName() << " is drawing a card." << endl;
        selectCard(enemy, cardArray);
        cout << " --------------------------" << endl;

         bool botChoice = true;

        while(user.calculateBestScore() < 21 && (answer != "stay" || botChoice))
        {

            if (answer!= "stay")
            {

                cout << " Your current score is " << user.calculateBestScore() << endl;
                cout << " Do you want to hit or stay?" << endl;
                cout << " Type 'hit' or 'stay'" << endl;
                cin >> answer;
            }


            while (answer != "hit" && answer != "stay")
            {
                cout << " fInvalid choice, select 'hit' or 'stay'" << endl;
                cin >> answer;
            }

            if (answer == "hit")
            {
                cout << " Drawing new card" << endl;
                selectCard(user, cardArray);
                user.printHand();
                cout << " Current Score: " << user.calculateBestScore() << endl;
            }

            if (botChoice != false)
            {
                botChoice = botSelection(enemy);
            }

            if(botChoice)
            {
                cout<< enemy.getName() << " is choosing to hit." << endl;
                selectCard(enemy, cardArray);
```



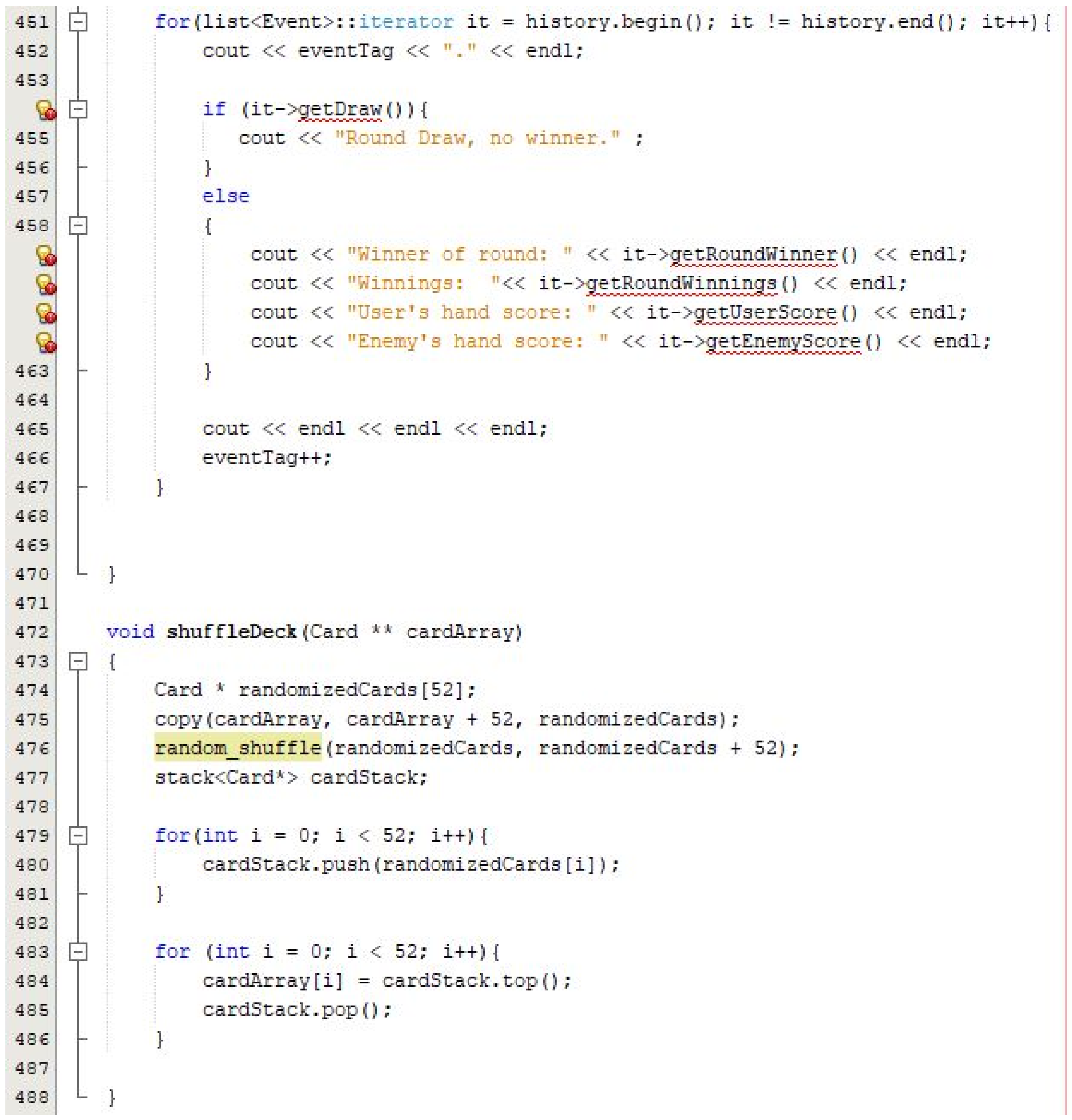
```cpp
                cout << "Player " << user.getName() << " has " << user.getTotalChipCount() << " chips" << endl;
                cout << enemy.getName() << " has " << enemy.getTotalChipCount() << " chips" << endl;
                gameHistory.push_back(Event(user, enemy, bet, 1, false));
            }

            else if (finalScoreUser <= 21 && finalScoreEnemy > 21){
                cout << " ---------------------------------------------" << endl;
                cout << "The winner of this round is"  << enemy.getName() << endl;
                cout << " ---------------------------------------------" << endl;
                enemy.updateChipWin(bet);
                user.updateChipLost(bet);
                cout << "New Chip amounts: " << endl;
                cout << "Player " << user.getName() << " has " << user.getTotalChipCount() << " chips" << endl;
                cout << enemy.getName() << " has " << enemy.getTotalChipCount() << " chips" << endl;
                gameHistory.push_back(Event(user, enemy, bet, 0, false));
            }

            else {

                int userDistanceFrom21 = abs(21 - finalScoreUser);
                int enemyDistanceFrom21 = abs(21 - finalScoreEnemy);

                if (userDistanceFrom21 < enemyDistanceFrom21){
                    cout << "------------------------------" << endl;
                    cout << user.getName() << " is the winner" << endl;
                    cout << "------------------------------" << endl;
                    user.updateChipWin(bet);
                    enemy.updateChipLost(bet);
                    cout << "New Chip amounts: " << endl;
                    cout << "Player " << user.getName() << " has " << user.getTotalChipCount() << " chips" << endl;
                    cout << enemy.getName() << " has " << enemy.getTotalChipCount() << " chips" << endl;
                    gameHistory.push_back(Event(user, enemy, bet, 1, false));
                }

                else if (userDistanceFrom21 > enemyDistanceFrom21){
                    cout << "------------------------------" << endl;
                    cout << enemy.getName() << " is the winner"<< endl;
                    cout << "------------------------------" << endl;
                    enemy.updateChipWin(bet);
                    user.updateChipLost(bet);
                    cout << "New Chip amounts: " << endl;
                    cout << "Player " << user.getName() << " has " << user.getTotalChipCount() << " chips" << endl;
                    cout << enemy.getName() << " has " << enemy.getTotalChipCount() << " chips" << endl;
                    gameHistory.push_back(Event(user, enemy, bet, 0, false));

                }

                else if (userDistanceFrom21 == enemyDistanceFrom21)
                {
                    cout << "------------------------------" << endl;
                    cout <<"Draw.  Both players have the same score" << endl;
                    cout << "------------------------------" << endl;
```



```cpp
    cardArray[randomCard]->setTaken();
    return;

}

bool botSelection(Player & enemy)
    {
    int choice = false;
    if(enemy.calculateBestScore() < 10){
        return true;
    }

    else if(enemy.calculateBestScore() >  10 && enemy.calculateBestScore() <= 16){
        choice = rand()%2;
        if(choice)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    else
    {
        return false;
    }

}

void printInstructions()
{
    cout << " -------------" << endl;
    cout << " How To Play: " << endl;
    cout << " -------------" << endl << endl;
    cout << " The goal of the game is to have the hand that has a value of 21 or has the closest value to 21." << endl;
    cout << " The game is played with one or more 52 standard decks, however in this case the game will be played with only 1 deck." << endl;
    cout << " If your hand value is over 21 causes you to bust, and you will lose regardless of the other player's hand." << endl;
    cout << " Aces have a value of 1 or 11." << endl;
    cout << " Cards 2-9 all have their designated number as their value." << endl;
    cout << " 10s, Jacks, Queens, and Kings all have a value of 10." << endl;
    cout << " A blackjack is the most powerful hand, consisting of an Ace and any card that has a value of 10." << endl;
    cout << " Each player is given 2 cards before anything is done." << endl;
    cout << " There are 3 different choices the player can make after the cards has been dealt: Hit, Stay, or Double Down." << endl;
    cout << " Hit: When you ask for another card, you can keep doing this till you are satisfied or till you bust." << endl;
    cout << " Stay: When you stay with your current hand and don't take in any extra cards." << endl;
    cout << " Double Down: When you double your bet and you take only 1 extra card." << endl;
}

int returnBetAmount(Player & user, Player & enemy)
{
```

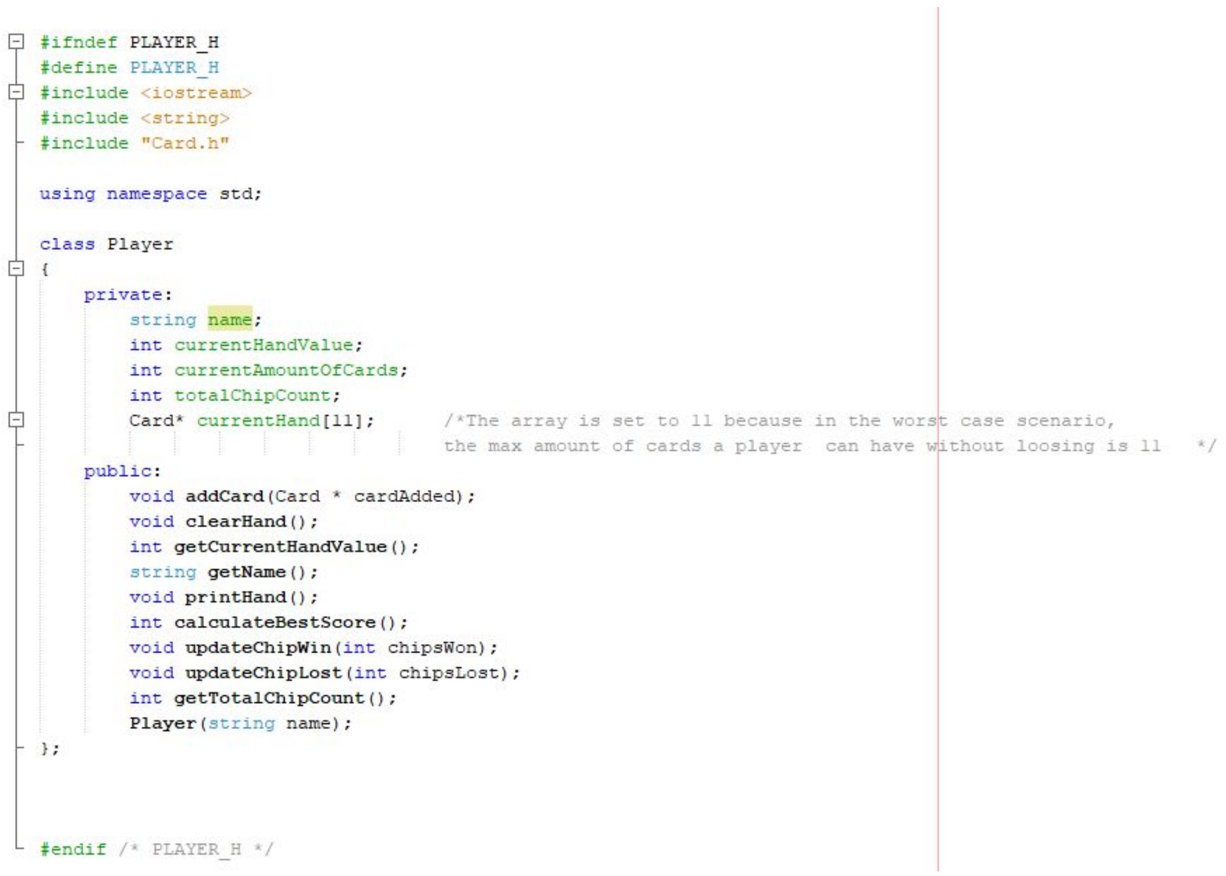


```
    cout << "You currently have " << user.getTotalChipCount() << " chips." << endl;
    cout << enemy.getName() << " currently has " << enemy.getTotalChipCount() << endl;
    cout << "Enter your bid amount.  Minimum 10 chips and must be divisible by 10: " << endl << endl;
    int ans;
    cin >> ans;

    while ((ans >= 0 && ans < 10) || (ans > user.getTotalChipCount()) || (ans > enemy.getTotalChipCount()) || (ans %10) )
    {
        cout << "INVALID CHIP AMOUNT." << endl;
        if(ans >= 0 && ans < 10)
        {
            cout << "Bet too low, minimum bet is 10 chips.  Enter valid chip amount: " << endl;
            cin >> ans;
        }
        else if(ans > user.getTotalChipCount())
        {
            cout << "Bet too high. You only have " << user.getTotalChipCount() << " chips. Enter valid chip amount: " << endl;
            cin >> ans;
        }
        else if(ans > enemy.getTotalChipCount())
        {
            cout << "Bet too high. " << enemy.getName() << " only has " << user.getTotalChipCount() << " chips. Enter valid chip amount: " << endl;
            cin >> ans;
        }
        else if (ans %10){
            cout << "Bet must be divisible by 10.  Enter valid chip amount: " << endl;
            cin >> ans;
        }
    }

    cout << "Betting " << ans << " chips." << endl;
    return ans;


}

void printGameHistory(list<Event> history)
{
     cout << endl << endl;
    cout << "-----------------------------" << endl;
    cout << "       Game History          " << endl;
    cout << "-----------------------------" << endl << endl << endl;

    if (history.begin() == history.end()){
        cout << "No game history, go play a game!" << endl << endl << endl ;
        return;
    }
    int eventTag = 1;


```



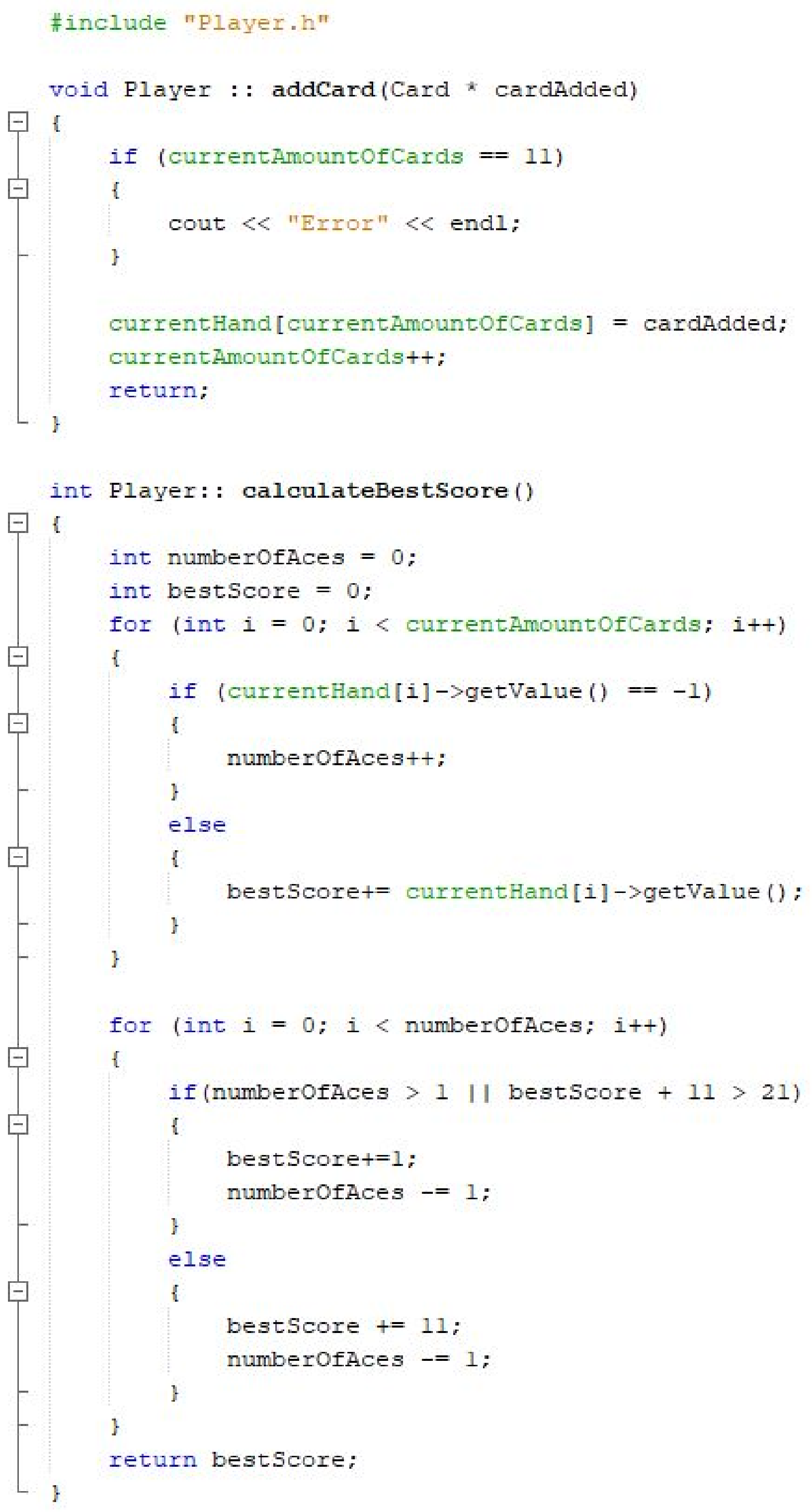
```
    for(list<Event>::iterator it = history.begin(); it != history.end(); it++){
        cout << eventTag << "." << endl;

        if (it->getDraw()){
           cout << "Round Draw, no winner." ;
        }
        else
        {
            cout << "Winner of round: " << it->getRoundWinner() << endl;
            cout << "Winnings:  "<< it->getRoundWinnings() << endl;
            cout << "User's hand score: " << it->getUserScore() << endl;
            cout << "Enemy's hand score: " << it->getEnemyScore() << endl;
        }

        cout << endl << endl << endl;
        eventTag++;
    }


}

void shuffleDeck(Card ** cardArray)
{
    Card * randomizedCards[52];
    copy(cardArray, cardArray + 52, randomizedCards);
    random_shuffle(randomizedCards, randomizedCards + 52);
    stack<Card*> cardStack;

    for(int i = 0; i < 52; i++){
        cardStack.push(randomizedCards[i]);
    }

    for (int i = 0; i < 52; i++){
        cardArray[i] = cardStack.top();
        cardStack.pop();
    }

}
```

Player.h

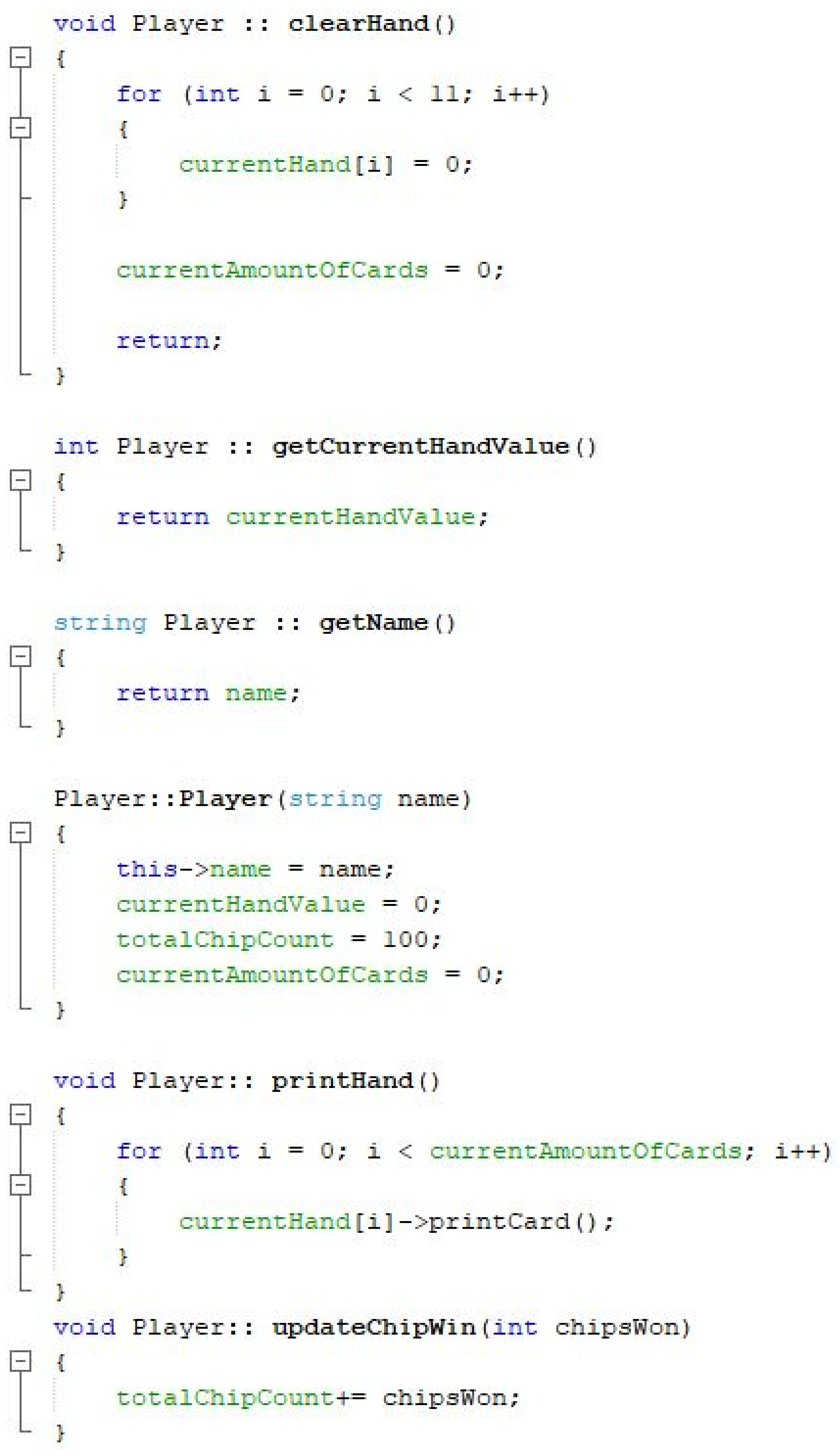


```
#ifndef PLAYER_H
#define PLAYER_H
#include <iostream>
#include <string>
#include "Card.h"

using namespace std;

class Player
{
    private:
        string name;
        int currentHandValue;
        int currentAmountOfCards;
        int totalChipCount;
        Card* currentHand[11];      /*The array is set to 11 because in the worst case scenario,
                                    the max amount of cards a player  can have without loosing is 11   */
    public:
        void addCard(Card * cardAdded);
        void clearHand();
        int getCurrentHandValue();
        string getName();
        void printHand();
        int calculateBestScore();
        void updateChipWin(int chipsWon);
        void updateChipLost(int chipsLost);
        int getTotalChipCount();
        Player(string name);
};



#endif /* PLAYER_H */
```

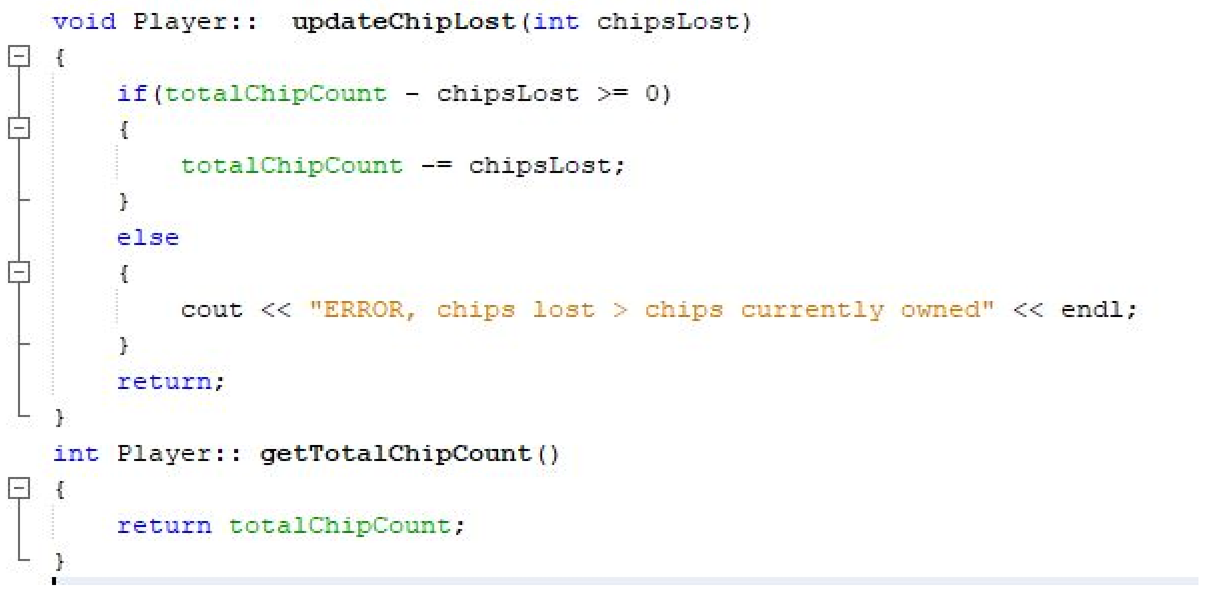
Player.cpp



```cpp
#include "Player.h"

void Player :: addCard(Card * cardAdded)
{
    if (currentAmountOfCards == 11)
    {
        cout << "Error" << endl;
    }

    currentHand[currentAmountOfCards] = cardAdded;
    currentAmountOfCards++;
    return;
}

int Player:: calculateBestScore()
{
    int numberOfAces = 0;
    int bestScore = 0;
    for (int i = 0; i < currentAmountOfCards; i++)
    {
        if (currentHand[i]->getValue() == -1)
        {
            numberOfAces++;
        }
        else
        {
            bestScore+= currentHand[i]->getValue();
        }
    }

    for (int i = 0; i < numberOfAces; i++)
    {
        if(numberOfAces > 1 || bestScore + 11 > 21)
        {
            bestScore+=1;
            numberOfAces -= 1;
        }
        else
        {
            bestScore += 11;
            numberOfAces -= 1;
        }
    }
    return bestScore;
}
```

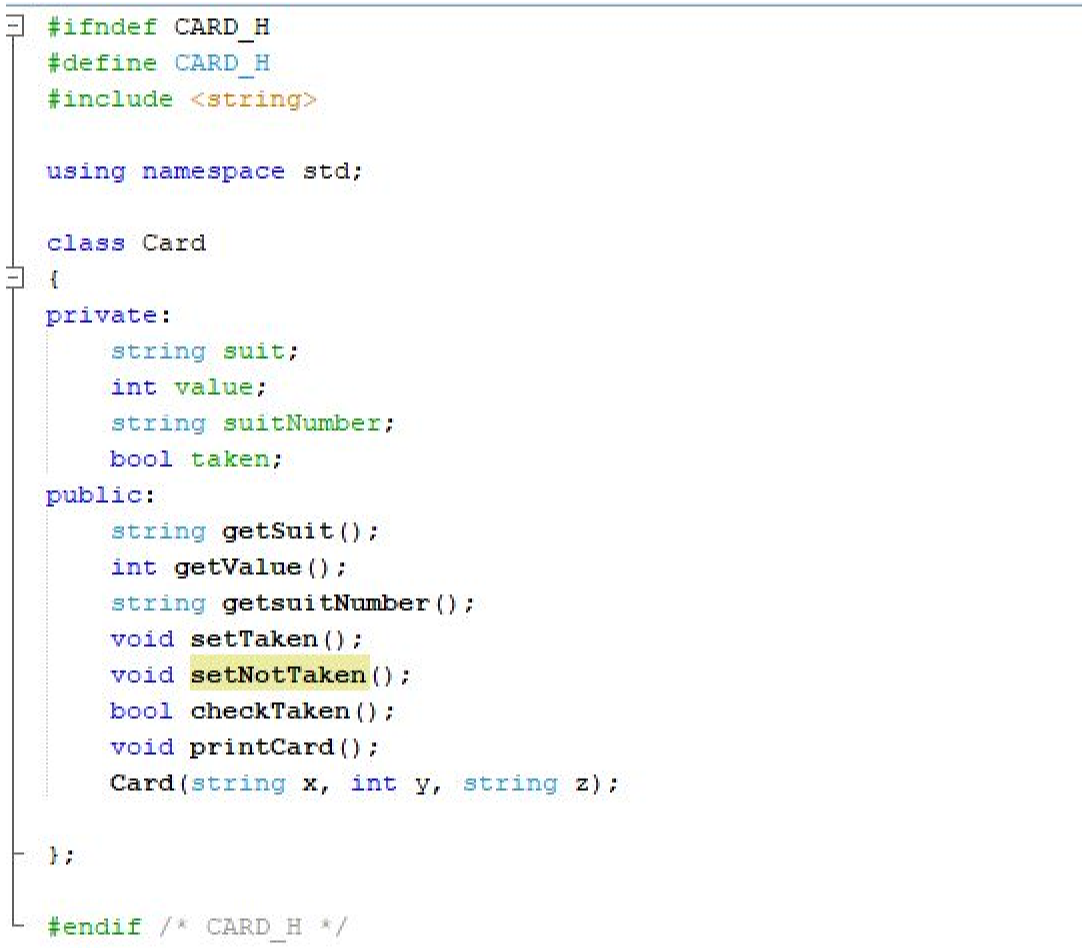


```
void Player :: clearHand()
{
    for (int i = 0; i < 11; i++)
    {
        currentHand[i] = 0;
    }

    currentAmountOfCards = 0;

    return;
}

int Player :: getCurrentHandValue()
{
    return currentHandValue;
}

string Player :: getName()
{
    return name;
}

Player::Player(string name)
{
    this->name = name;
    currentHandValue = 0;
    totalChipCount = 100;
    currentAmountOfCards = 0;
}

void Player:: printHand()
{
    for (int i = 0; i < currentAmountOfCards; i++)
    {
        currentHand[i]->printCard();
    }
}
void Player:: updateChipWin(int chipsWon)
{
    totalChipCount+= chipsWon;
}
```



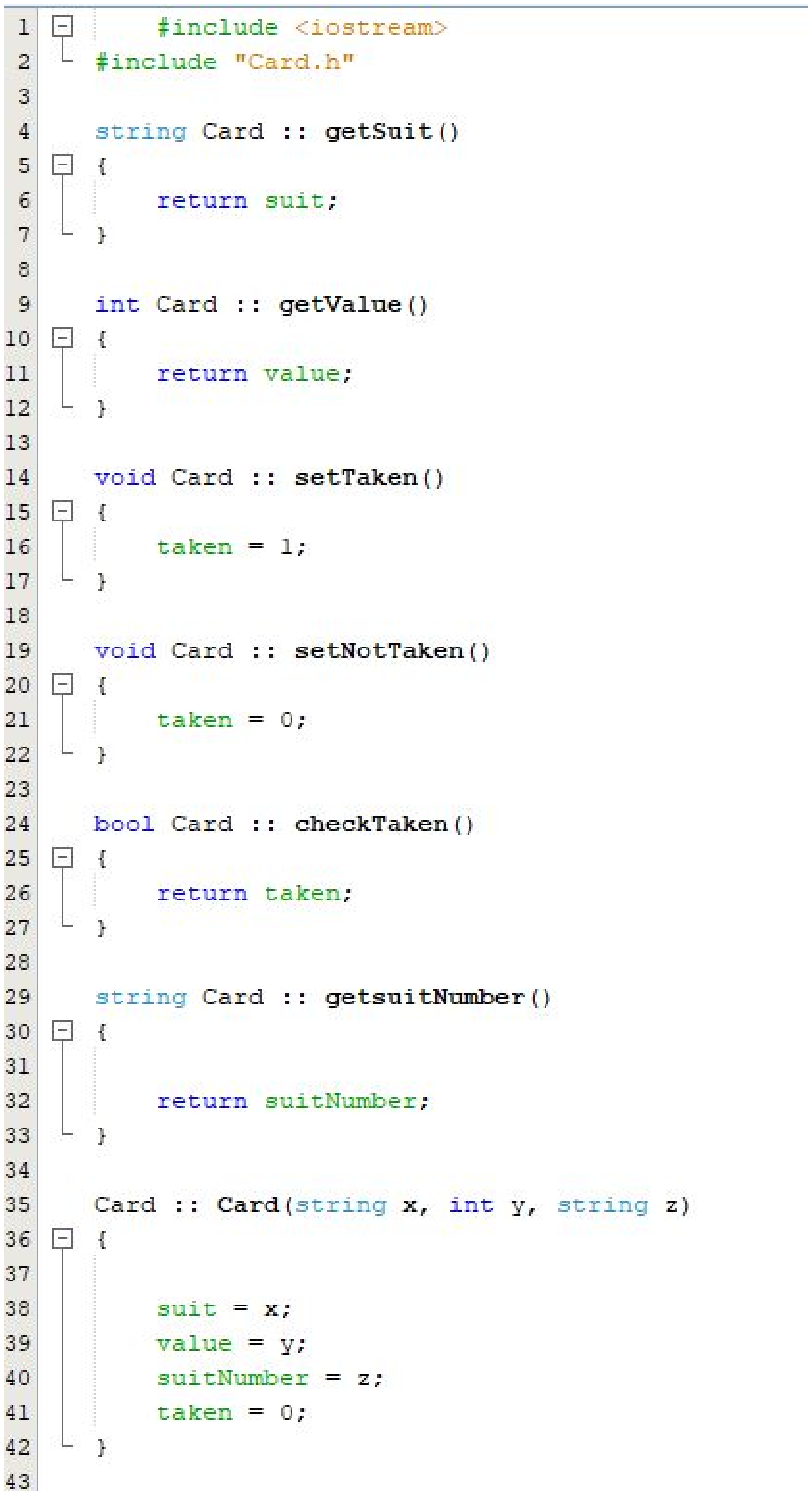
```cpp
void Player::  updateChipLost(int chipsLost)
{
    if(totalChipCount - chipsLost >= 0)
    {
        totalChipCount -= chipsLost;
    }
    else
    {
        cout << "ERROR, chips lost > chips currently owned" << endl;
    }
    return;
}
int Player:: getTotalChipCount()
{
    return totalChipCount;
}
```

Card.h

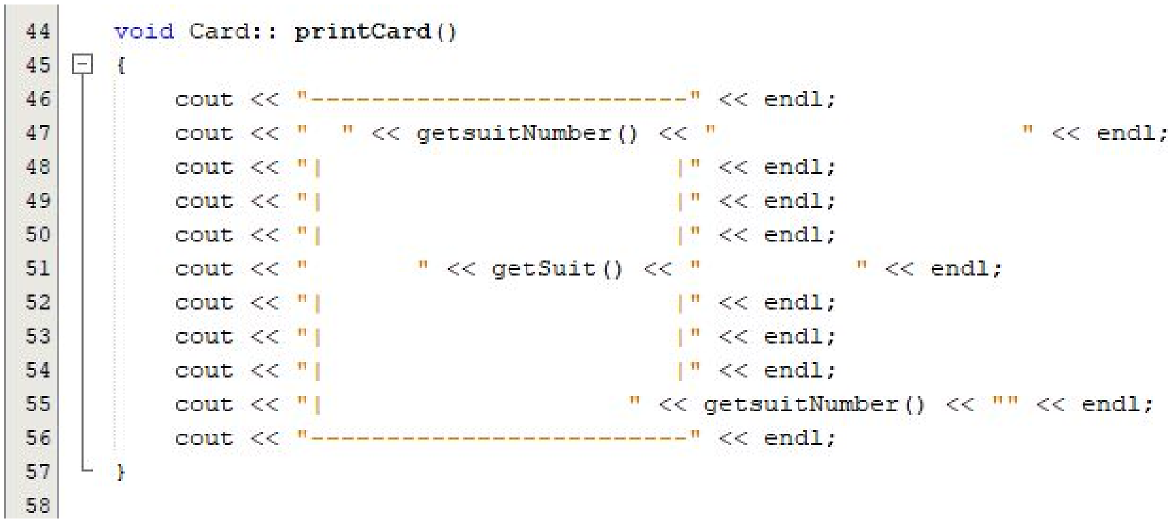


```cpp
#ifndef CARD_H
#define CARD_H
#include <string>

using namespace std;

class Card
{
private:
    string suit;
    int value;
    string suitNumber;
    bool taken;
public:
    string getSuit();
    int getValue();
    string getsuitNumber();
    void setTaken();
    void setNotTaken();
    bool checkTaken();
    void printCard();
    Card(string x, int y, string z);

};

#endif /* CARD_H */
```

Card.cpp



```cpp
    #include <iostream>
#include "Card.h"

string Card :: getSuit()
{
    return suit;
}

int Card :: getValue()
{
    return value;
}

void Card :: setTaken()
{
    taken = 1;
}

void Card :: setNotTaken()
{
    taken = 0;
}

bool Card :: checkTaken()
{
    return taken;
}

string Card :: getsuitNumber()
{

    return suitNumber;
}

Card :: Card(string x, int y, string z)
{

    suit = x;
    value = y;
    suitNumber = z;
    taken = 0;
}

```

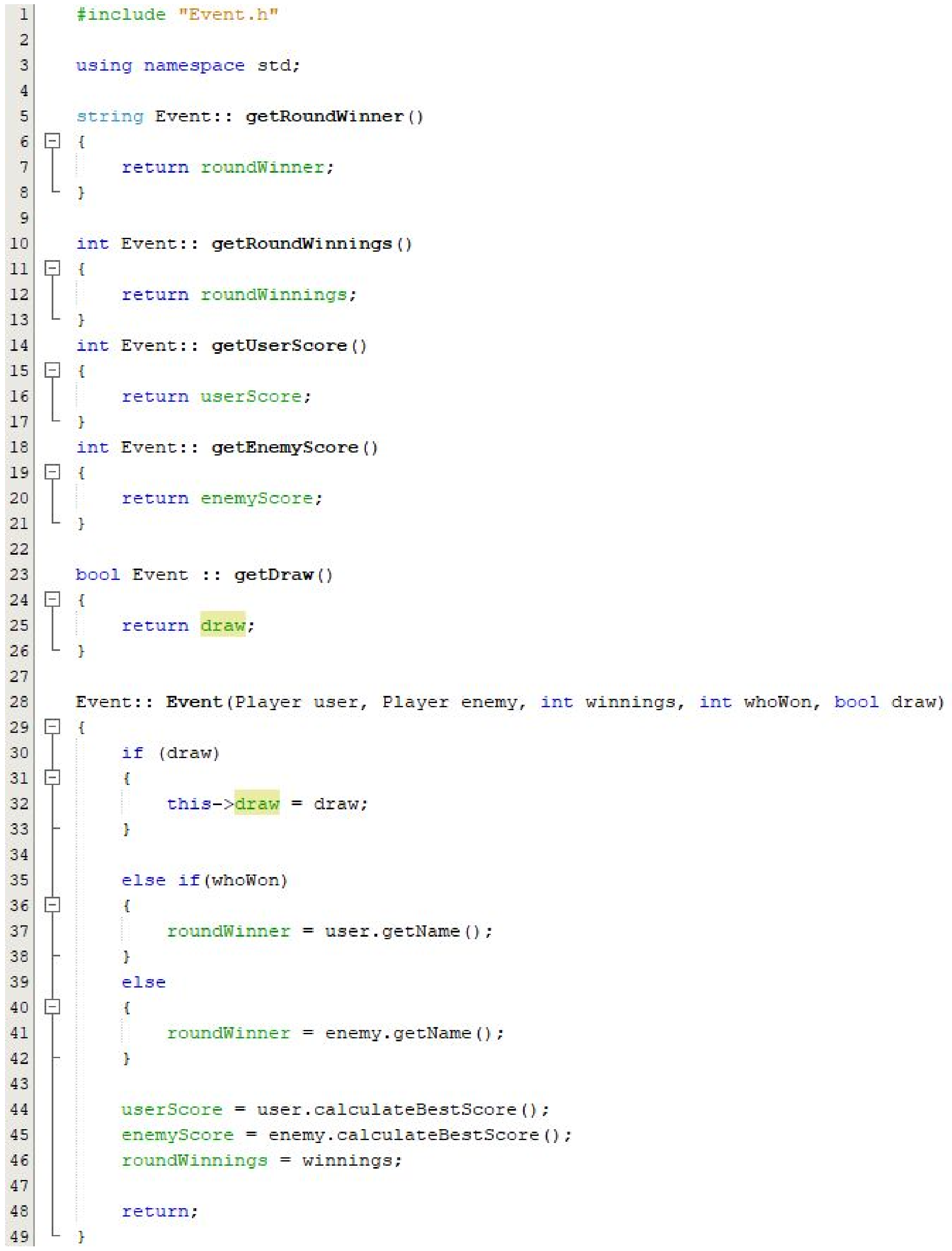


```cpp
void Card:: printCard()
{
    cout << "-------------------------" << endl;
    cout << "  " << getsuitNumber() << "                    " << endl;
    cout << "|                       |" << endl;
    cout << "|                       |" << endl;
    cout << "|                       |" << endl;
    cout << "       " << getSuit() << "          " << endl;
    cout << "|                       |" << endl;
    cout << "|                       |" << endl;
    cout << "|                       |" << endl;
    cout << "|                    " << getsuitNumber() << "" << endl;
    cout << "-------------------------" << endl;
}
```

Event.h

```cpp
#ifndef EVENT_H
#define EVENT_H
#include <string>
#include "Player.h"

using namespace std;

class Event{
private:
    string roundWinner;
    int roundWinnings;
    int userScore;
    int enemyScore;
    bool draw;

public:
    string getRoundWinner();
    int getRoundWinnings();
    int getUserScore();
    int getEnemyScore();
    bool getDraw();
    Event(Player user, Player enemy, int winnings, int whoWon, bool draw);
};


#endif /* EVENT_H */
```

Event.cpp



```cpp
#include "Event.h"

using namespace std;

string Event:: getRoundWinner()
{
    return roundWinner;
}

int Event:: getRoundWinnings()
{
    return roundWinnings;
}
int Event:: getUserScore()
{
    return userScore;
}
int Event:: getEnemyScore()
{
    return enemyScore;
}

bool Event :: getDraw()
{
    return draw;
}

Event:: Event(Player user, Player enemy, int winnings, int whoWon, bool draw)
{
    if (draw)
    {
        this->draw = draw;
    }

    else if(whoWon)
    {
        roundWinner = user.getName();
    }
    else
    {
        roundWinner = enemy.getName();
    }

    userScore = user.calculateBestScore();
    enemyScore = enemy.calculateBestScore();
    roundWinnings = winnings;

    return;
}
```