Table Of Contents

What I Intend to Share:

I intend to share all my source files with the class.

Design:

The goal in this project was to implement a 5-card draw poker game that follows all the standard rules. In the final showdown of the game, a previous assignment was combined with this program to compare the final hands to determine a winner. This previous assignment was heavily modified to be compatible with the current program and ensure proper functionality. Player objects are created and processed within the pokerGame objects. A player can either be an AI, or a player, so an abstract class was created for a player so that it simplified the game logic within the pokerGame class. The current AI is simple and mostly random with some bias in certain decisions, I considered using another abstract class for making decisions such that the AI logic could also be swapped out easily, but this felt beyond the scope of the project.

Source Code:

Main

```
#include <iostream>
#include "PokerGame.h"

using namespace std;

int main()
{
	PokerGame mainGame;

	mainGame.initGame();
	mainGame.createDeck();
	mainGame.shuffleDeck();
	mainGame.playPoker();

	return 0;
}
```

Player

```
#pragma once

#include <vector>
#include <string>
#include <iostream>
#include <memory>
#include <numeric>
#include <algorithm>
#include <sstream>
#include <thread>
#include <chrono>

#include "Card.h"

class PlayerI
{
public:
	std::string name;
	int moneyBet;
	std::vector<CardClass::Card> hand;
	bool isDone;
	bool isFolded;

	virtual void lookAtHand() = 0;
	virtual int placeBet(int previousBet, bool& bettingIsOpen) = 0;
	virtual void drawCards(std::vector<CardClass::Card> deck) = 0;

};
```

AiPlayer

```h
#pragma once

#include "PlayerI.h"

class AiPlayer : public PlayerI
{
public:
	AiPlayer(int index);
	int placeBet(int previousBet, bool& bettingIsOpen) override;
	void drawCards(std::vector<CardClass::Card> deck) override;
	void lookAtHand() override;

private:

	int getHandStrength();
};
```

```cpp
#include "AiPlayer.h"

AiPlayer::AiPlayer(int index)
{
	this->name = "Ai Player " + std::to_string(index);
    moneyBet = 0;
    isDone = false;
    isFolded = false;
}

int AiPlayer::placeBet(int previousBet, bool& bettingIsOpen)
{
    std::cout << "The opponent studies their hand..." << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(6000));

    // AI opens the round if the previous bet was 0
    if (!bettingIsOpen)
    {
        // Randomly check or bet
        int randomChoice = rand() % 2;
        if (randomChoice == 0)
        {
            std::cout << this->name << " checks." << std::endl;
            return 0;
        }
        else
        {
            // Place a random starting bet between 5 and 10
            int randomBet = 5 + 1 + (rand() % 6);
            moneyBet += randomBet;
            bettingIsOpen = true;
            return randomBet;
        }
    }
```

```
    int handStrength = getHandStrength();

    // Decide the bet based on hand strength
    int bet = 0;
    int randomChoice = rand() % 5;

    if (handStrength > 60)
    {
        // Randomly choose between a aggressive raise and a call
        if (0 <= randomChoice && randomChoice <= 2)// Call the previous bet
        {
            bet = previousBet;
        }
        else// Aggressive bet
        {
            bet = previousBet * 2;
        }
    }
    else if (handStrength > 30)// Moderate Bet
    {
        // Randomly choose between a moderate raise and a call
        if (0 <= randomChoice && randomChoice <= 3)// Call the previous bet
        {
            bet = previousBet;
        }
        else// Moderate bet
        {
            bet = previousBet * 1.3;
        }
    }
    else
    {
        bet = 0;
    }

    moneyBet += bet;
    return bet;
}

void AiPlayer::drawCards(std::vector<CardClass::Card> deck)
{
    // Ensure the deck has enough cards to draw from
    if (deck.empty())
    {
        std::cout << "Deck is empty! Cannot replace cards." << std::endl;
        return;
    }

    // Delay so it feels more like a game
    std::cout << "The opponent studies their hand..." << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(3000));

    // Randomly choose number of cards to discard
    int cardsToReplace = std::rand() % 6;
    if (cardsToReplace == 0)
    {
        std::cout << this->name << " decides to keep all their cards." <<
std::endl;
        return;
    }
```

```
    // Select random indices from the AI's hand
    std::vector<int> indices(hand.size());
    std::iota(indices.begin(), indices.end(), 0);
    std::random_shuffle(indices.begin(), indices.end());

    // Replace the selected cards
    for (int i = 0; i < cardsToReplace; i++)
    {
        int index = indices[i];

        if (!deck.empty())
        {
            hand[index] = deck.back();
            deck.pop_back();
        }
        else
        {
            std::cout << "Not enough cards in the deck to replace all cards!"
<< std::endl;
            break;
        }
    }

    std::cout << this->name << " replaced " << cardsToReplace << " cards." <<
std::endl;
}

void AiPlayer::lookAtHand()
{
    std::cout << this->name << "'s hand: " << std::endl;

    for (const auto& card : hand)
    {
        std::cout << card.rank << " of " << card.suit << ", ";
    }

    std::cout << std::endl << std::endl;
}

int AiPlayer::getHandStrength()
{
    int strength = 0;
    for (const auto& card : hand)
    {
        if (card.rank == "A" || card.rank == "K" || card.rank == "Q" ||
card.rank == "J") {
            strength += 20;
        }
        else
        {
            strength += std::stoi(card.rank);
        }
    }

    return std::min(strength, 100);
}
```

HumanPlayer

```
#pragma once

#include "PlayerI.h"

class HumanPlayer : public PlayerI
{
public:
	HumanPlayer(std::string name);
	int placeBet(int previousBet, bool& bettingIsOpen) override;
	void drawCards(std::vector<CardClass::Card> deck) override;
	void lookAtHand() override;
	void lookAtHandIndices();

};
#include "HumanPlayer.h"

HumanPlayer::HumanPlayer(std::string name)
{
	  this->name = name;
    moneyBet = 0;
    isDone = false;
    isFolded = false;
}

int HumanPlayer::placeBet(int previousBet, bool& bettingIsOpen)
{
    lookAtHand();

    if (bettingIsOpen)
    {
        std::cout << "The current bet is $" << previousBet << std::endl;
        std::cout << "You can choose to:" << std::endl;
        std::cout << "1. Call (match the bet)" << std::endl;
        std::cout << "2. Raise (bet more)" << std::endl;
        std::cout << "3. Fold (exit the round)" << std::endl;
    }
    else
    {
        std::cout << "No bets have been placed yet. You can:" << std::endl;
        std::cout << "1. Check" << std::endl;
        std::cout << "2. Bet a starting amount (Ante is: $" << previousBet <<
")" << std::endl;
    }

    int choice = 0;
    int betAmount = 0;

    while (true)
    {
        std::cout << "Enter your choice: ";
        std::cin >> choice;
        if (choice == 1)
        {
            // Call or check
            if (bettingIsOpen)
            {
                moneyBet += previousBet;
```

```
                return previousBet;
            }
            else
            {
                std::cout << "You checked." << std::endl;
                return 0;
            }
        }
        else if (choice == 2)
        {
            // Raise or place a starting bet
            if (bettingIsOpen)
            {
                std::cout << "Enter the amount to raise: ";
            }
            else
            {
                std::cout << "Enter your starting bet: ";
            }

            std::cin >> betAmount;
            if (betAmount > previousBet )
            {
                if (!bettingIsOpen)
                {
                    bettingIsOpen = true;
                }
                moneyBet += betAmount;
                return betAmount;
            }
            else
            {
                std::cout << "Invalid bet. Try again." << std::endl;
            }
        }
        else if (choice == 3 && bettingIsOpen)
        {
            // Fold
            isFolded = true;
            return 0;
        }
        else
        {
            std::cout << "Invalid choice. Try again." << std::endl;
        }
    }
}


void HumanPlayer::drawCards(std::vector<CardClass::Card> deck)
{
    std::cout << this->name << "'s turn to draw new cards!" << std::endl;

    lookAtHandIndices();

    std::cout << "Enter the indices of the cards you want to replace (space-
separated).";
    std::cout << "Press Enter without inputting anything to keep all cards:
";
```

```
    // Clear the input buffer only if necessary
    if (std::cin.peek() == '\n')
    {
        std::cin.ignore();
    }

    std::string input;
    std::getline(std::cin, input);

    if (input.empty())
    {
        std::cout << "No cards will be replaced." << std::endl;
        return;
    }

    std::vector<int> indicesToReplace;
    std::istringstream iss(input);
    int index;

    while (iss >> index)
    {
        if (index >= 0 && static_cast<size_t>(index) < hand.size())
        {
            indicesToReplace.push_back(index);
        }
        else
        {
            std::cout << "Invalid index: " << index << ". Skipping." <<
std::endl;
        }
    }

    // Replace the selected cards
    for (int indexx : indicesToReplace)
    {
        if (!deck.empty()) {
            hand[indexx] = deck.back();
            deck.pop_back();
        }
        else
        {
            std::cout << "Deck is empty! Cannot draw more cards." <<
std::endl;
            break;
        }
    }

    // Display the updated hand
    std::cout << std::endl << this->name << "'s updated hand:" << std::endl;
    for (size_t i = 0; i < hand.size(); i++)
    {
        std::cout << i << ": " << hand[i].rank << " of " << hand[i].suit <<
std::endl;
    }

    std::cout << std::endl;
}

void HumanPlayer::lookAtHand()
{
```

```
    std::cout << this->name << "'s hand: " << std::endl;

    for (const auto& card : hand)
    {
        std::cout << card.rank << " of " << card.suit << ", ";
    }

    std::cout << std::endl << std::endl;
}

void HumanPlayer::lookAtHandIndices()
{
    std::cout << this->name << "'s hand:" << std::endl;

    for (size_t i = 0; i < hand.size(); ++i)
    {
        std::cout << i << ": " << hand[i].rank << " of " << hand[i].suit <<
std::endl;
    }

    std::cout << std::endl;
}
```

Card

```
#pragma once

#include <string>

class CardClass
{
public:
      struct Card {
            std::string rank;
            std::string suit;
      };
};
```

PokerGame

```
#pragma once

#include <vector>
#include <memory>
#include <string>
#include <iostream>
#include <algorithm>

#include "Card.h"
#include "PlayerI.h"
#include "HumanPlayer.h"
#include "AiPlayer.h"
#include "pokerHand.h"

class PokerGame
{
public:
      PokerGame();
```

```
        void shuffleDeck();
        void initGame();
        void playPoker();
        void createDeck();

private:
        std::vector<CardClass::Card> deck;
        std::vector<std::shared_ptr<PlayerI>> players;
        const int MAX_NUM_PLAYERS = 7;
        const int MIN_NUM_PLAYERS = 2;
        int moneyPot;
        int previousBet;
        int ante;
        bool bettingIsOpen;

        void dealCards();
        void showDown();
        bool checkFoldedWinner();
        void bettingRound();
        void drawRound();
        int getValidUserInt();

};
```

```
#include "PokerGame.h"

void PokerGame::createDeck()
{
    const std::vector<std::string> ranks = { "2", "3", "4", "5", "6", "7",
"8", "9", "10", "J", "Q", "K", "A" };
    const std::vector<std::string> suits = { "Hearts", "Diamonds", "Clubs",
"Spades" };

    for (const auto& rank : ranks)
    {
        for (const auto& suit : suits)
        {
            this->deck.push_back({ rank, suit });
        }
    }
}

PokerGame::PokerGame(){
    moneyPot = 0;
    previousBet = 0;
    ante = 0;
    bettingIsOpen = false;
}

void PokerGame::shuffleDeck()
{
    // Seed the rand function so its different everytime
    std::srand(static_cast<unsigned int>(std::time(nullptr)));

    for (size_t i = this->deck.size() - 1; i > 0; i++)
    {
        int j = std::rand() % (i + 1); // Generate a random index
        std::swap(this->deck[i], this->deck[j]);// Swap the current element
with the randomly chosen element
    }
```

```
}

void PokerGame::initGame()
{
    // Get umber of human players (0 - 7)
    int numHuman;
    do
    {
        std::cout << "Enter valid number of human players: ";
        numHuman = getValidUserInt();
    } while ( numHuman > MAX_NUM_PLAYERS );

    // Get number of AI players (2 <= max players <= 7)
    int numAi;
    do
    {
        std::cout << "Enter valid number of AI players: ";
        numAi = getValidUserInt();
    } while ( (numHuman + numAi) > MAX_NUM_PLAYERS || (numHuman + numAi) <
MIN_NUM_PLAYERS );

    // Create all player objects
    std::string name;

    for (int i = 0; i < numHuman; i++)
    {
        std::cout << "Enter player " << i << "s name: ";
        std::getline(std::cin, name);
        players.push_back(std::make_shared<HumanPlayer>(name));
    }

    for (int i = 0; i < numAi; i++)
    {
        players.push_back(std::make_shared<AiPlayer>(i+1));
    }

    // New player to start each time
    std::random_shuffle(players.begin(), players.end());

    std::cout << "Whats the ante?: ";
    ante = getValidUserInt();

    moneyPot += (numAi + numHuman) * ante;
    std::cout << "The pot contains $" << moneyPot << std::endl;
}

void PokerGame::playPoker()
{
    // Deal 5 cards to each player
    dealCards();

    // First round of betting
    previousBet = ante;
    bettingIsOpen = false;
    bettingRound();

    // See if all but one person folded
    if (checkFoldedWinner()) {
        return;
    }
```

```
    // Draw Round
    drawRound();

    // Fresh betting round, per standard rules
    previousBet = ante;
    bettingIsOpen = false;
    bettingRound();

    // Its showtime
    showDown();
}

void PokerGame::dealCards()
{
    std::cout << std::endl << "Dealing 5 cards to each player." << std::endl
<< std::endl;

    for (auto& player : players)
    {
        for (int i = 0; i < 5; ++i)
        {
            player->hand.push_back(this->deck.back());
            this->deck.pop_back();
        }
    }
}

void PokerGame::showDown()
{
    if (players.empty())
    {
        std::cout << "No players to create hands for." << std::endl;
        return;
    }

    // Create a vector to store pokerHand objects and track active (non-
folded) players
    std::vector<pokerHand> activeHands;
    std::vector<std::shared_ptr<PlayerI>> activePlayers;

    for (const auto& player : players)
    {
        if (!player->isFolded)
        {
            activeHands.emplace_back(player->hand);
            activePlayers.push_back(player); // Track the player
corresponding to the hand
        }
    }

    std::cout << "It's showtime." << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(3000));
    std::cout << "Everyone drops their cards..." << std::endl;
    std::this_thread::sleep_for(std::chrono::milliseconds(6000));

    // Determine the best hand and the corresponding player
    size_t bestIndex = 0;

    for (size_t i = 1; i < activeHands.size(); i++)
```

```
    {
        if (activeHands[i].compare(activeHands[bestIndex]) == 1)
        {
            bestIndex = i;
        }
    }

    std::cout << "The winner is: " << activePlayers[bestIndex]->name << "!"
<< std::endl;
    activePlayers[bestIndex]->lookAtHand();
    std::cout << "Which is a " << activeHands[bestIndex].getHandRankString()
<< std::endl << std::endl;
    std::cout << "Pot won: $ " << moneyPot << std::endl << std::endl;
    std::cout << "Player Stats: " << std::endl;

    for (const auto& player : players)
    {
        std::cout << player->name << std::endl << "Money bet: $" << player-
>moneyBet << std::endl;
        player->lookAtHand();
    }

    std::cout << std::endl << "Thanks for playing!" << std::endl;
}

bool PokerGame::checkFoldedWinner()
{
    PlayerI* remainingPlayer = nullptr;

    for (auto& player : players)
    {
        if (!player->isFolded)
        {
            if (remainingPlayer != nullptr)
            {
                return false;
            }
            remainingPlayer = player.get();
        }
    }

    if (remainingPlayer != nullptr)
    {
        std::cout << remainingPlayer->name << " wins the hand!
Congratulations!" << std::endl;
        return true;
    }

    return false;
}

void PokerGame::bettingRound()
{
    std::cout << "Begin the betting round!" << std::endl << std::endl;

    // First reset the status of all players
    for (auto& player : players)
    {
        player->isDone = false;
    }
```

```
    bool bettingDone = false;

    while (!bettingDone)
    {
        bettingDone = true;

        for (auto& player : players)
        {
            if (!player->isDone && !player->isFolded)
            {
                std::cout << player->name << "'s turn to place a bet!" <<
std::endl;
                int bet = player->placeBet(previousBet, bettingIsOpen);

                if (bet > previousBet)
                {
                    std::cout << player->name << " raises with $" << bet <<
std::endl;
                    previousBet = bet;

                    // Since someone raised, give everyone a chance to play
again, except for the person who just raised
                    for (auto& remainingPlayer : players)
                    {
                        remainingPlayer->isDone = false;
                    }

                    player->isDone = true;
                }
                else if (bet == 0 && bettingIsOpen)
                {
                    std::cout << player->name << " folds." << std::endl;
                    player->isFolded = true;
                }
                else if (bet == previousBet)
                {
                    std::cout << player->name << " calls." << std::endl;
                    player->isDone = true;
                }

                moneyPot += bet; // Add valid bets to the pot
                std::cout << std::endl << "The pot contains: $" << moneyPot
<< std::endl << std::endl;
            }
        }

        // Edge case for if everyone checks continuously
        if (!bettingIsOpen)
        {
            bettingDone = false;
        }

        // Check if the round is over
        for (auto& player : players)
        {
            if (!player->isDone && !player->isFolded)
            {
                bettingDone = false;
            }
```

```
        }
    }

    std::cout << "The betting round is over!" << std::endl << std::endl;
}

void PokerGame::drawRound()
{
    std::cout << "Begin the draw round!" << std::endl << std::endl;

    for (auto& player : players)
    {
        if (!player->isFolded)
        {
            player->drawCards(this->deck);
        }
    }

    std::cout << "The draw round is over!" << std::endl << std::endl;
}

int PokerGame::getValidUserInt()
{
    int value;
    while (true)
    {
        std::cin >> value;

        // Check if input is valid
        if (std::cin.fail() || value < 0) {
            std::cin.clear(); // Clear error flag
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n'); // Discard invalid input
            std::cout << "Invalid input. Please enter a valid integer." <<
std::endl;
        }
        else {
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n'); // Discard extra input
            return value;
        }
    }
}
```

Test driver:

The test was to test at least one combination of Ai players and human players. Multiple games are shown below.

Results:

The program plays poker, with a random playing AI!

Game 1 (0 Human, 3 Ai):

```
Enter valid number of human players: 0
```

Enter valid number of AI players: 3

Whats the ante?: 5

The pot contains $15

Dealing 5 cards to each player.

Begin the betting round!

Ai Player 1's turn to place a bet!

The opponent studies their hand...

Ai Player 1 checks.

The pot contains: $15

Ai Player 2's turn to place a bet!

The opponent studies their hand...

Ai Player 2 checks.

The pot contains: $15

Ai Player 3's turn to place a bet!

The opponent studies their hand...

Ai Player 3 checks.

The pot contains: $15

Ai Player 1's turn to place a bet!

The opponent studies their hand...

Ai Player 1 raises with $7

The pot contains: $22

Ai Player 2's turn to place a bet!

The opponent studies their hand...

Ai Player 2 folds.

The pot contains: $22

Ai Player 3's turn to place a bet!

The opponent studies their hand...

Ai Player 3 calls.

The pot contains: $29

The betting round is over!

Begin the draw round!

The opponent studies their hand...

Ai Player 1 decides to keep all their cards.

The opponent studies their hand...

Ai Player 3 replaced 2 cards.

The draw round is over!

Begin the betting round!

Ai Player 1's turn to place a bet!

The opponent studies their hand...

Ai Player 1 checks.

The pot contains: $29

Ai Player 3's turn to place a bet!

The opponent studies their hand...

Ai Player 3 raises with $11

The pot contains: $40

Ai Player 1's turn to place a bet!

The opponent studies their hand...

Ai Player 1 calls.

The pot contains: $51

The betting round is over!

It's showtime.

Everyone drops their cards...

The winner is: Ai Player 1!

Ai Player 1's hand:

3 of Spades, K of Clubs, 2 of Clubs, 4 of Spades, 2 of Diamonds,

Which is a One pair

Pot won: $ 51

Player Stats:

Ai Player 1

Money bet: $18

Ai Player 1's hand:

3 of Spades, K of Clubs, 2 of Clubs, 4 of Spades, 2 of Diamonds,

Ai Player 2

Money bet: $0

Ai Player 2's hand:

3 of Clubs, 10 of Hearts, 8 of Spades, 4 of Clubs, 5 of Diamonds,

Ai Player 3

Money bet: $18

Ai Player 3's hand:

K of Spades, J of Spades, 6 of Clubs, 4 of Hearts, A of Hearts,

Thanks for playing!

C:\Users\timfe\Documents\MASTERS\C++\PokerGame\PokerGame\x64\Debug\PokerGame.exe (process 18608) exited with code 0 (0x0).

Press any key to close this window . . .

Game 2 (1 Human, 3 Ai):

Enter valid number of human players: 1

Enter valid number of AI players: 3

Enter player 0s name: Tim

Whats the ante?: 5

The pot contains $20

Dealing 5 cards to each player.

Begin the betting round!

Tim's turn to place a bet!

Tim's hand:

J of Spades, J of Clubs, 10 of Spades, 4 of Clubs, 6 of Hearts,

No bets have been placed yet. You can:

1. Check

2. Bet a starting amount (Ante is: $5)

Enter your choice: 2

Enter your starting bet: 6

Tim raises with $6

The pot contains: $26

Ai Player 1's turn to place a bet!

The opponent studies their hand...

Ai Player 1 calls.

The pot contains: $32

Ai Player 3's turn to place a bet!

The opponent studies their hand...

Ai Player 3 calls.

The pot contains: $38

Ai Player 2's turn to place a bet!

The opponent studies their hand...

Ai Player 2 calls.

The pot contains: $44

The betting round is over!

Begin the draw round!

Tim's turn to draw new cards!

Tim's hand:

0: J of Spades

1: J of Clubs

2: 10 of Spades

3: 4 of Clubs

4: 6 of Hearts

Enter the indices of the cards you want to replace (space-separated).Press Enter without inputting anything to keep all cards: 3 4

Tim's updated hand:

0: J of Spades

1: J of Clubs

2: 10 of Spades

3: 7 of Clubs

4: 10 of Diamonds

The opponent studies their hand...

Ai Player 1 replaced 1 cards.

The opponent studies their hand...

Ai Player 3 replaced 2 cards.

The opponent studies their hand...

Ai Player 2 decides to keep all their cards.

The draw round is over!

Begin the betting round!

Tim's turn to place a bet!

Tim's hand:

J of Spades, J of Clubs, 10 of Spades, 7 of Clubs, 10 of Diamonds,

No bets have been placed yet. You can:

1. Check

2. Bet a starting amount (Ante is: $5)

Enter your choice: 2

Enter your starting bet: 20

Tim raises with $20

The pot contains: $64

Ai Player 1's turn to place a bet!

The opponent studies their hand...

Ai Player 1 raises with $26

The pot contains: $90

Ai Player 3's turn to place a bet!

The opponent studies their hand...

Ai Player 3 calls.

The pot contains: $116

Ai Player 2's turn to place a bet!

The opponent studies their hand...

Ai Player 2 raises with $33

The pot contains: $149

Tim's turn to place a bet!

Tim's hand:

J of Spades, J of Clubs, 10 of Spades, 7 of Clubs, 10 of Diamonds,

The current bet is $33

You can choose to:

1. Call (match the bet)

2. Raise (bet more)

3. Fold (exit the round)

Enter your choice: 2

Enter the amount to raise: 35

Tim raises with $35

The pot contains: $184

Ai Player 1's turn to place a bet!

The opponent studies their hand...

Ai Player 1 raises with $45

The pot contains: $229

Ai Player 3's turn to place a bet!

The opponent studies their hand...

Ai Player 3 calls.

The pot contains: $274

Ai Player 2's turn to place a bet!

The opponent studies their hand...

Ai Player 2 calls.

The pot contains: $319

Tim's turn to place a bet!

Tim's hand:

J of Spades, J of Clubs, 10 of Spades, 7 of Clubs, 10 of Diamonds,

The current bet is $45

You can choose to:

1. Call (match the bet)

2. Raise (bet more)

3. Fold (exit the round)

Enter your choice: 1

Tim calls.

The pot contains: $364

The betting round is over!

It's showtime.

Everyone drops their cards...

The winner is: Ai Player 1!

Ai Player 1's hand:

K of Clubs, 7 of Clubs, K of Hearts, 3 of Hearts, 7 of Diamonds,

Which is a Two pair

Pot won: $ 364

Player Stats:

Tim

Money bet: $106

Tim's hand:

J of Spades, J of Clubs, 10 of Spades, 7 of Clubs, 10 of Diamonds,

Ai Player 1

Money bet: $77

Ai Player 1's hand:

K of Clubs, 7 of Clubs, K of Hearts, 3 of Hearts, 7 of Diamonds,

Ai Player 3

Money bet: $77

Ai Player 3's hand:

A of Hearts, 10 of Diamonds, 10 of Clubs, 2 of Clubs, 7 of Clubs,

Ai Player 2

Money bet: $84

Ai Player 2's hand:

2 of Hearts, 3 of Clubs, Q of Clubs, 6 of Spades, K of Spades,

Thanks for playing!

C:\Users\timfe\Documents\MASTERS\C++\PokerGame\PokerGame\x64\Debug\PokerGame.exe (process 23184) exited with code 0 (0x0).

Press any key to close this window . . .

Game 3 (2 Human, 1 Ai):

Enter valid number of human players: 2

Enter valid number of AI players: 1

Enter player 0s name: Tim

Enter player 1s name: Evil Tim

Whats the ante?: 5

The pot contains $15

Dealing 5 cards to each player.

Begin the betting round!

Tim's turn to place a bet!

Tim's hand:

3 of Clubs, 10 of Spades, Q of Hearts, 4 of Hearts, 4 of Diamonds,

No bets have been placed yet. You can:

1. Check

2. Bet a starting amount (Ante is: $5)

Enter your choice: 1

You checked.

The pot contains: $15

Evil Tim's turn to place a bet!

Evil Tim's hand:

7 of Clubs, 8 of Spades, 6 of Clubs, A of Spades, 8 of Clubs,

No bets have been placed yet. You can:

1. Check

2. Bet a starting amount (Ante is: $5)

Enter your choice: 2

Enter your starting bet: 6

Evil Tim raises with $6

The pot contains: $21

Ai Player 1's turn to place a bet!

The opponent studies their hand...

Ai Player 1 raises with $12

The pot contains: $33

Tim's turn to place a bet!

Tim's hand:

3 of Clubs, 10 of Spades, Q of Hearts, 4 of Hearts, 4 of Diamonds,

The current bet is $12

You can choose to:

1. Call (match the bet)

2. Raise (bet more)

3. Fold (exit the round)

Enter your choice: 1

Tim calls.

The pot contains: $45

Evil Tim's turn to place a bet!

Evil Tim's hand:

7 of Clubs, 8 of Spades, 6 of Clubs, A of Spades, 8 of Clubs,

The current bet is $12

You can choose to:

1. Call (match the bet)

2. Raise (bet more)

3. Fold (exit the round)

Enter your choice: 1

Evil Tim calls.

The pot contains: $57

The betting round is over!

Begin the draw round!

Tim's turn to draw new cards!

Tim's hand:

0: 3 of Clubs

1: 10 of Spades

2: Q of Hearts

3: 4 of Hearts

4: 4 of Diamonds

Enter the indices of the cards you want to replace (space-separated).Press Enter without inputting anything to keep all cards: 0 1 2

Tim's updated hand:

0: J of Clubs

1: Q of Spades

2: 6 of Spades

3: 4 of Hearts

4: 4 of Diamonds

Evil Tim's turn to draw new cards!

Evil Tim's hand:

0: 7 of Clubs

1: 8 of Spades

2: 6 of Clubs

3: A of Spades

4: 8 of Clubs

Enter the indices of the cards you want to replace (space-separated).Press Enter without inputting anything to keep all cards: 0 2

Evil Tim's updated hand:

0: J of Clubs

1: 8 of Spades

2: Q of Spades

3: A of Spades

4: 8 of Clubs

The opponent studies their hand...

Ai Player 1 replaced 1 cards.

The draw round is over!

Begin the betting round!

Tim's turn to place a bet!

Tim's hand:

J of Clubs, Q of Spades, 6 of Spades, 4 of Hearts, 4 of Diamonds,

No bets have been placed yet. You can:

1. Check

2. Bet a starting amount (Ante is: $5)

Enter your choice: 2

Enter your starting bet: 20

Tim raises with $20

The pot contains: $77

Evil Tim's turn to place a bet!

Evil Tim's hand:

J of Clubs, 8 of Spades, Q of Spades, A of Spades, 8 of Clubs,

The current bet is $20

You can choose to:

1. Call (match the bet)

2. Raise (bet more)

3. Fold (exit the round)

Enter your choice: 1

Evil Tim calls.

The pot contains: $97

Ai Player 1's turn to place a bet!

The opponent studies their hand...

Ai Player 1 raises with $40

The pot contains: $137

Tim's turn to place a bet!

Tim's hand:

J of Clubs, Q of Spades, 6 of Spades, 4 of Hearts, 4 of Diamonds,

The current bet is $40

You can choose to:

1. Call (match the bet)

2. Raise (bet more)

3. Fold (exit the round)

Enter your choice: 1

Tim calls.

The pot contains: $177

Evil Tim's turn to place a bet!

Evil Tim's hand:

J of Clubs, 8 of Spades, Q of Spades, A of Spades, 8 of Clubs,

The current bet is $40

You can choose to:

1. Call (match the bet)

2. Raise (bet more)

3. Fold (exit the round)

Enter your choice: 1

Evil Tim calls.

The pot contains: $217

The betting round is over!

It's showtime.

Everyone drops their cards...

The winner is: Evil Tim!

Evil Tim's hand:

J of Clubs, 8 of Spades, Q of Spades, A of Spades, 8 of Clubs,

Which is a One pair

Pot won: $ 217

Player Stats:

Tim

Money bet: $72

Tim's hand:

J of Clubs, Q of Spades, 6 of Spades, 4 of Hearts, 4 of Diamonds,

Evil Tim

Money bet: $78

Evil Tim's hand:

J of Clubs, 8 of Spades, Q of Spades, A of Spades, 8 of Clubs,

Ai Player 1

Money bet: $52

Ai Player 1's hand:

J of Clubs, K of Spades, 2 of Clubs, 10 of Diamonds, 9 of Spades,

Thanks for playing!

C:\Users\timfe\Documents\MASTERS\C++\PokerGame\PokerGame\x64\Debug\PokerGame.exe (process 14944) exited with code 0 (0x0).

Press any key to close this window . . ■