OOP With Design Pattern

Student Name: Bichi Zhang

ID#: 1073838

Program3

Due Date: 03/06/2019

Date of Submission: 03/06/2019

**Table of Contents**

Documented Source Code…………………………………………………………………………………………………….3

Screenshots ………………………………………………………………………………………………………………………….8

Container Justification and Complexity Analysis……………………………………………………………………8

Things I Learned…………………………………………………………………………………………………………………….8

Documented Source Code:

File Name: main.cpp

Author: Bichi Zhang

Date: 03/01/2019

Complier Used: C++ by Visual Studio 2017

Description: In this file, use the domain function to get the first 5 cards for each player and use one\_player\_hand to get the next two rounds for each player.

void one\_player\_hand(std::string name, Deck& deck) {

Card cards[4];

std::cout << name << ": ";

for (int i = 0;i < 4; ++i) {

cards[i] = deck.hand\_one();

std::cout << cards[i] << " ";

}

std::cout << std::endl;

}

void domain() {

Deck deck;

deck.simulate\_shuffling();

std::string players[4] = {"North", "East", "South", "West"};

std::cout << "round 1:" << std::endl;

for (int i = 0;i < 4; ++i) {

std::cout << players[i] << ": ";

for (int j = 0;j < 5; ++j) {

std::cout << deck.hand\_one() << " ";

}

std::cout << std::endl;

}

int round = 2;

while (!deck.is\_empty()) {

std::cout << "round " << round++ << ":" << std::endl;

for (int i = 0;i < 4; ++i) {

one\_player\_hand(players[i], deck);

}

}

}

int main() {

std::string input;

while (true) {

domain();

std::cout << "repeat? (y/n) ";

std::cin >> input;

if (input != "y") {

break;

}

}

return 0;

}

File Name: deck. h

Author: Bichi Zhang

Date: 03/02/2019

Complier Used: C++ by Visual Studio 2017

Description: In this header file, I name the function of class card and deck.

#include <string>

#include <iostream>

#include <list>

class Card {

public:

Card() {}

Card(std::string c, std::string v):

color(c), value(v) {

}

friend std::ostream& operator<<(std::ostream& out, const Card& rhs);

private:

std::string color;

std::string value;

};

class Deck {

public:

Deck();

void reset();

void simulate\_shuffling();

Card hand\_one();

void restore(Card& card);

bool is\_empty() { return cards.empty(); }

private:

std::list<Card> cards;

};

File Name: deck.cpp

Author: Bichi Zhang

Date: 03/05/2019

Complier Used: C++ by Visual Studio 2017

Description: In this file, I overload the operator “<<” to output the suit and value of card. I use the deck::reset to initialize the first deck and use the simulate\_shuffling function to shuffle it. I also define a function called hand\_one to give one player a card each time.

#include "deck.h"

#include <random>

#include <algorithm>

#include <vector>

#include <time.h>

std::ostream& operator<<(std::ostream& out, const Card& rhs) {

out << rhs.color << "-" << rhs.value;

return out;

}

void Deck::reset() {

std::string colors[4] = {"Club", "Diamond", "Heart", "Spade"};

std::string values[13] = {"Ace", "King", "Queen", "Jack", "10", "9", "8", "7", "6", "5", "4", "3", "2"};

for (int i = 0;i < 4; ++i) {

for (int j = 0;j < 13; ++j) {

Card card(colors[i], values[j]);

cards.push\_back(card);

}

}

}

Deck::Deck() {

reset();

}

void Deck::simulate\_shuffling() {

std::vector<Card> vect;

for (std::list<Card>::iterator it = cards.begin();it != cards.end(); ++it) {

vect.push\_back(\*it);

}

std::random\_shuffle(vect.begin(), vect.end());

std::list<Card> new\_cards;

for (std::vector<Card>::iterator it = vect.begin();it != vect.end(); ++it) {

new\_cards.push\_back(\*it);

}

cards = new\_cards;

}

Card Deck::hand\_one() {

Card card = cards.front();

cards.pop\_front();

return card;

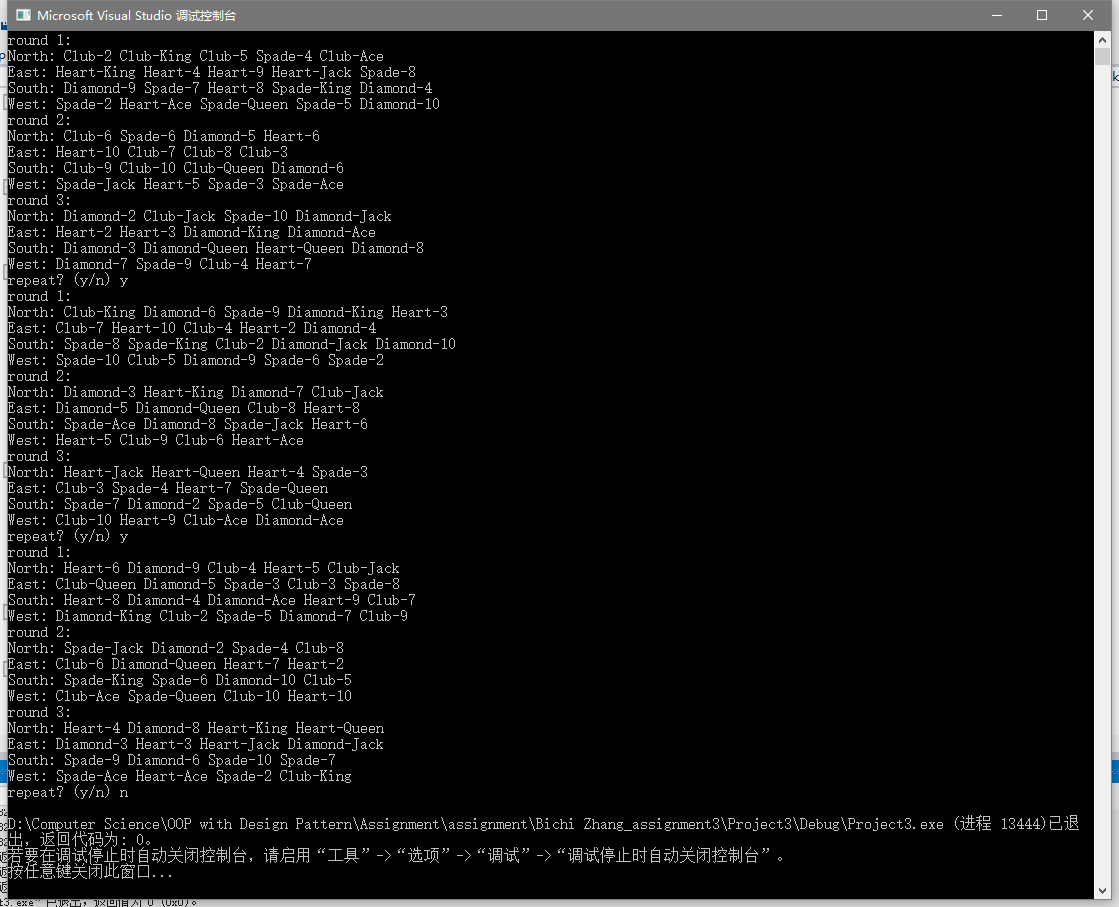
}

void Deck::restore(Card& card) {

cards.push\_back(card);

}

Screenshot:



Contain Justification And Complexity Analysis:

In this project, I use the list cards to store a deck of cards. For vector, elements can be added to or removed from the end in constant time, but insertion and removal from the beginning and the middle are linear-time operations. The list template class represents a doubly linked list. Each element, other than the first and last, is linked to the item before it and the item following it, implying that a list can be traversed in both direction. We need to initialize a deck of cards and simulates shuffling it and dealing a bridge hand. List provides for constant-time insertion and removal of elements at any location in the list. So it will be a good choice to use list rather than vector.

Things I learned:

In the process of completing assignment 3, I realized that STL is really a powerful tool. In this assignment, I used “push\_back”, “front”, “pop\_front”, “random\_shuffle” to achieve my goal. With the help of STL, the code is relatively small this time. I have reason to believe that STL is important in future learning and work.