c++大作业报告

问题描述：

“21点”是一个古老的扑克牌游戏，游戏规则是：各个参与者设法使自己的牌达到总分21而不超过这个数值。扑克牌的分值取它们的面值，A充当1分或者11分（由玩家自己选择一种分值），J.Q和K人头牌都是10分。庄家对付1到7个玩家。在一局开始时，包括庄家在内的所有参与者都有两张牌。玩家可以看到他们的所有牌以及总分，而庄家有一张牌暂时是隐蔽的。接下来，只有愿意，各个玩家都有机会依次再拿一张牌。如果某个玩家的总分超过了21（称为“引爆”），那么这个玩家就输了。在所有玩家都拿了额外的牌后，庄家将显示隐藏的牌。只要庄家的总分小于17，那么他就必须再拿牌，如果庄家引爆了，那么还没有引爆的所有玩家都将获胜。引爆的玩家打成平局。否则，将余下的各玩家的总分与庄家的总分做比较，如果玩家的总分大于庄家的总分，则玩家获胜。如果二者的总分相同，则玩家与庄家打成平局。

设计要求：

1）玩家可以任意添牌（此阶段玩家爆牌即输掉筹码）

2）玩家添牌结束后，庄家亮出隐藏牌。庄家点数<17必须添牌，>=17必须停牌（庄家爆牌，输；否则，比大小）

3）不需要做图形界面；支持多人游戏

结构设计：

1）类设计

Card类：

成员	描述
rank m_Rank	rank: 枚举类型
suit m_Suit	suit: 枚举类型
bool m_IsFaceUp	标示牌是否正面朝上的状态
int GetValue()	返回牌面值所对应的分值
void Flip()	翻面

Hand类：

成员	描述
vector<Card*> m_Cards	存储指向Card对象指针
void Add(Card* pCard)	添加一张牌
void Clear()	清除手中所有牌
int GetTotal() const	返回这首派总值

GenericPlayer类：

成员	描述
string m_Name	名字
virtual bool IsHitting() const=0	指出通用玩家是否想要另一张牌
bool IsBusted() const	是否引爆
void Bust() const	显示

Player类：

成员	描述
string m_Name	
virtual bool IsHitting() const=0	
bool IsBusted() const	判断是否爆牌
void Bust() const	显示爆牌
void Win() const	宣布玩家获胜
void Lose() const	宣布玩家失败
void Draw() const	宣布玩家获得平局
	总资产
m_Bet	筹码

House类：

成员	描述
string m_Name	
virtual bool IsHitting() const=0	指出庄家是否再拿一张牌
bool IsBusted() const	
void Bust() const	显示爆牌
void FlipFirstCard()	翻转第一张牌

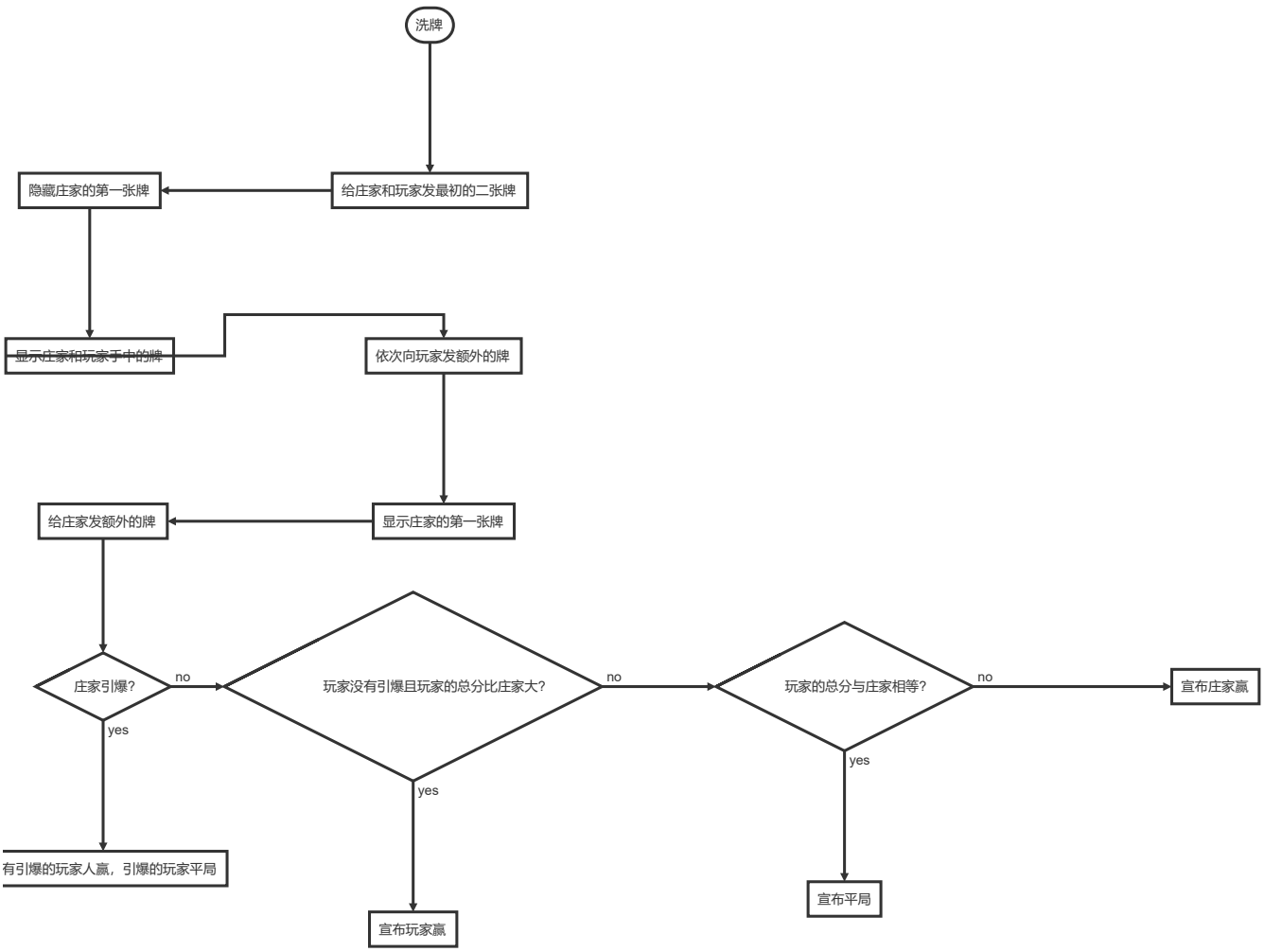
Deck类：

成员	描述
void Populate()	创建一共52张的标准扑克牌
void Shuffle()	洗牌
void Deal(Hand& aHand)	将一张牌发到一手中
void AdditionalCards(GenericPlayer* aGenericPlayer)	发牌

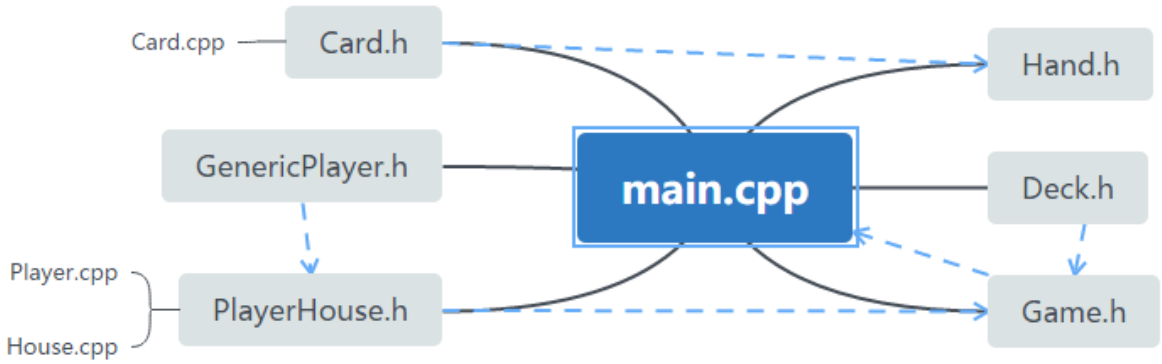
Game类：

成员	描述
Deck m_Deck	一副牌
House m_House	庄家
Player m_Players[10]	普通玩家的集合
void Play()	玩一局游戏
Menu()	显示菜单
Save()	保存当前设置
Continue()	继续上次游戏
InitUser()	初始化单人玩家
InitUsers()	初始化多人玩家
JudgePlay()	判断还要不要再来一局
Help()	帮助

2）游戏流程


洗牌
给庄家和玩家发最初的二张牌
隐藏庄家的第一张牌
显示庄家和玩家手中的牌
依次向玩家发额外的牌
显示庄家的第一张牌
给庄家发额外的牌
庄家引爆?
no
yes
玩家没有引爆且玩家的总分比庄家大?
no
yes
玩家的总分与庄家相等?
no
yes
宣布庄家赢
有引爆的玩家人赢，引爆的玩家平局
宣布玩家赢
宣布平局

3）框图设计


Card.cpp
Card.h
Hand.h
GenericPlayer.h
main.cpp
Deck.h
Player.cpp
House.cpp
PlayerHouse.h
Game.h

测试示例：

C:\Users\xzw\Desktop\oop\main.exe

```
------------------------
Welcome to xzw's 21 points!
 Menu
0:Continue
1:Single Game
2:Multiplayer Game
3:Help
4:Exit
------------------------
Your choice:2
Welcome to xzw's 21 points!
How many people are going to play?(1-7):3
No.1, please enter your name:xzw
How many dollars do you have?1000
No.2, please enter your name:xyn
How many dollars do you have?500
No.3, please enter your name:yzt
How many dollars do you have?100
Game begin!
xzw ,how many dollars do you want to bet?
100
xyn ,how many dollars do you want to bet?
200
yzt ,how many dollars do you want to bet?
200
xzw:    DiamondsK       Clubs3  Total:13

xyn:    Hearts7 HeartsA Total:18

yzt:    DiamondsQ       Diamonds7       Total:17

House:  XX      ClubsK

xzw, do you want a hit? (Y/N): y
xzw:    DiamondsK       Clubs3  Spades4 Total:17

xzw, do you want a hit? (Y/N): y
xzw:    DiamondsK       Clubs3  Spades4 Hearts5 Total:22

xzw busts.

xyn, do you want a hit? (Y/N): n

yzt, do you want a hit? (Y/N): y
yzt:    DiamondsQ       Diamonds7       ClubsJ  Total:27

yzt busts.

House:  Diamonds9       ClubsK  Total:19
```

```
House:  Diamonds9       ClubsK  Total:19

xzw You Lose!
You have 900 dollars now.
xyn You Lose!
You have 300 dollars now.
yzt You Lose!
You have -100 dollars now.
yzt , you have owed XZW(the house) a lot!
Once again?(Y/N)
n
------------------------
Welcome to xzw's 21 points!
 Menu
0:Continue
1:Single Game
2:Multiplayer Game
3:Help
4:Exit
------------------------
Your choice:3
21 piont is an ancient poker game. The rules of the game are:
each player tries to make his card score 21 but not more than that.
Poker cards are valued at their face value. A acts as a score of 1
or 11 (a score chosen by the player himself). J. Q and K are both 10
points. The dealer deals with one to seven players. At the beginning
of the game, all participants, including the banker, had two cards.
Players can see all their cards and total points, while the dealer has
 a card that is temporarily hidden. Next, each player has the opportunity
 to take another card in turn if he wants to. If a players total score
exceeds 21, the player loses. After all players have taken extra cards,
the dealer will show hidden cards. As long as the bankers total score
is less than 17, then he must take another card. If the banker detonates,
 then all players who have not detonated will win. The player who
detonated was tied. Otherwise, compare the total score of the remaining
players with the total score of the banker. If the total score of the
player is greater than the total score of the banker, the player wins.
If the total score of the two is the same, the player and the banker draw.
------------------------
Welcome to xzw's 21 points!
 Menu
0:Continue
1:Single Game
2:Multiplayer Game
3:Help
4:Exit
------------------------
Your choice:
```

源码：

Card.h

```
#ifndef _CARD_H_
#define _CARD_H_

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

```
#include <ctime>
using namespace std;

class Card
{
public:
    enum rank {ACE = 1, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE, TEN,
               JACK, QUEEN, KING};
    enum suit {CLUBS, DIAMONDS, HEARTS, SPADES};

    //overloading << operator so can send Card object to standard output
    friend ostream& operator<<(ostream& os, const Card& aCard);

    Card(rank r = ACE, suit s = SPADES, bool ifu = true);

    //returns the value of a card, 1 - 11
    int GetValue() const;

    //flips a card; if face up, becomes face down and vice versa
    void Flip();

private:
    rank m_Rank;
    suit m_Suit;
    bool m_IsFaceUp;
};
#endif
```

Card.cpp

```
#include "Card.h"
Card::Card(rank r, suit s, bool ifu) : m_Rank(r), m_Suit(s), m_IsFaceUp(ifu)
{
}

int Card::GetValue() const
{
    //if a card is face down, its value is 0
    int value = 0;
    if (m_IsFaceUp)
    {
        //value is number showing on card
        value = m_Rank;
        //value is 10 for face cards
        if (value > 10)
        {
            value = 10;
        }
    }
    return value;
}

void Card::Flip()
```

```
{
    m_IsFaceUp = !(m_IsFaceUp);
}
```

Hand.h

```
#ifndef _HAND_H_
#define _HAND_H_
#include "Card.h"
#include <vector>
using namespace std;
class Hand
{
public:
  Hand();

  virtual ~Hand();
  void Add(Card *pCard);
  void Clear();
  int GetTotal() const;

protected:
  vector<Card *> m_Cards;
};
#endif
```

Hand.cpp

```
#include "Hand.h"

Hand::Hand()
{
    m_Cards.reserve(7);
}

Hand::~Hand()
{
    Clear();
}

void Hand::Add(Card *pCard)
{
    m_Cards.push_back(pCard);
}

void Hand::Clear()
{
    vector<Card *>::iterator iter = m_Cards.begin();

    for (iter = m_Cards.begin(); iter != m_Cards.end(); ++iter)
    {
        delete *iter;
```

```
        *iter = 0;
    }
    m_Cards.clear();
}

int Hand::GetTotal() const
{
    if (m_Cards.empty())
    {
        return 0;
    }
    if (m_Cards[0]->GetValue() == 0)
    {
        return 0;
    }
    int total = 0;
    vector<Card *>::const_iterator iter;
    for (iter = m_Cards.begin(); iter != m_Cards.end(); ++iter)
    {
        total += (*iter)->GetValue();
    }

    bool containsAce = false;

    for (iter = m_Cards.begin(); iter != m_Cards.end(); ++iter)
    {
        if ((*iter)->GetValue() == Card::ACE)
        {
            containsAce = true;
        }
    }
    if (containsAce && total <= 11)
    {
        total += 10;
    }
    return total;
}
```

GenericPlayer.h

```
#ifndef _GENERICPLAYER_H_
#define _GENERICPLAYER_H_
#include "Hand.h"
#include <string>
using namespace std;
class GenericPlayer : public Hand
{
public:
    friend ostream &operator<<(ostream &os, const GenericPlayer &aGenericPlayer);
    GenericPlayer(const string &name = " ");
    virtual ~GenericPlayer();
    virtual bool IsHitting() const = 0;
    bool IsBusted() const;
```

```
    void Bust() const;

protected:
    string m_Name;
};
#endif
```

GenericPlayer.cpp

```
#include "GenericPlayer.h"
GenericPlayer::GenericPlayer(const string &name) : m_Name(name)
{
}

GenericPlayer::~GenericPlayer()
{
}

bool GenericPlayer::IsBusted() const
{
    return (GetTotal() > 21);
}

void GenericPlayer::Bust() const
{
    cout << m_Name << " busts.\n";
}
```

PlayerHouse.h

```
#ifndef _PLAYERHOUSE_H_
#define _PLAYERHOUSE_H_
#include "GenericPlayer.h"
#include <iostream>
using namespace std;
class Player : public GenericPlayer
{
public:
    Player(const string &name = "");

    virtual ~Player();
    virtual bool IsHitting() const;
    void Win() const;
    void Lose() const;
    void Draw() const;

    int m_Money;
    int m_Bet;
};

class House : public GenericPlayer
{
```

```
public:
    House(const string &name = "House");
    virtual ~House();
    virtual bool IsHitting() const;
    void FlipFirstCard();
};
#endif
```

House.cpp

```
#include "PlayerHouse.h"
House::House(const string &name) : GenericPlayer(name)
{
}

House::~House()
{
}

bool House::IsHitting() const
{
    return (GetTotal() <= 16);
}

void House::FlipFirstCard()
{
    if (!(m_Cards.empty()))
    {
        m_Cards[0]->Flip();
    }
    else
    {
        cout << "No card to flip!\n";
    }
}
```

Player.cpp

```
#include "PlayerHouse.h"

Player::Player(const string &name) : GenericPlayer(name)
{
}

Player::~Player()
{
}

bool Player::IsHitting() const
{
    cout << m_Name << ", do you want a hit? (Y/N): ";
    char response;
```

```
    cin >> response;
    return (response == 'y' || response == 'Y');
}
void Player::Win() const
{

    cout << m_Name << " You win!" << endl;
    cout << "You have "<<m_Money<<" dollars now."<<endl;
}
void Player::Lose() const
{

    cout << m_Name << " You Lose!" << endl;
    cout << "You have "<<m_Money<<" dollars now."<<endl;
}
void Player::Draw() const
{
    cout << m_Name << " You draw!" << endl;
    cout << "You have "<<m_Money<<" dollars now."<<endl;
}
```

Deck.h

```
#ifndef _DECK_H_
#define _DECK_H_
#include "Hand.h"
#include "GenericPlayer.h"
#include <iostream>
using namespace std;
class Deck : public Hand
{
public:
    Deck();

    virtual ~Deck();
    void Populate();
    void Shuffle();
    void Deal(Hand &aHand);
    void AdditionalCards(GenericPlayer &aGenericPlayer);
};
#endif
```

Deck.cpp

```
#include "Deck.h"
#include <algorithm>
Deck::Deck()
{
    m_Cards.reserve(52);
    Populate();
}
```

```
Deck::~Deck()
{
}

void Deck::Populate()
{
    Clear();
    for (int s = Card::CLUBS; s <= Card::SPADES; ++s)
    {
        for (int r = Card::ACE; r <= Card::KING; ++r)
        {
            Add(new Card(static_cast<Card::rank>(r),
                         static_cast<Card::suit>(s)));
        }
    }
}

void Deck::Shuffle()
{
    random_shuffle(m_Cards.begin(), m_Cards.end());
}

void Deck::Deal(Hand &aHand)
{
    if (!m_Cards.empty())
    {
        aHand.Add(m_Cards.back());
        m_Cards.pop_back();
    }
    else
    {
        cout << "Out of cards. Unable to deal.";
    }
}

void Deck::AdditionalCards(GenericPlayer &aGenericPlayer)
{
    cout << endl;
    while (!(aGenericPlayer.IsBusted()) && aGenericPlayer.IsHitting())
    {
        Deal(aGenericPlayer);
        cout << aGenericPlayer << endl;

        if (aGenericPlayer.IsBusted())
        {
            aGenericPlayer.Bust();
        }
    }
}
```

Game.h

```
#ifndef _GAME_H_
```

```
#define _GAME_H_
#include "Deck.h"
#include "PlayerHouse.h"
#include <string>
#include <vector>
#include <iostream>
#include<windows.h>
#include<time.h>
#include<fstream>
#include<iomanip>
using namespace std;
class Game
{
public:
    Game();
    ~Game();

    int Menu();
    void Save();
    void Continue();
    void InitUser();
    void InitUsers();
    void Play();
    int JudgePlay();
    void Help();

    Deck m_Deck;
    House m_House;
    Player m_Players[10];

};
#endif
```

Game.cpp

```
#include "Game.h"
int n;

Game::Game()
{
    n = 0;
    srand(static_cast<unsigned int>(time(0)));
    m_Deck.Populate();
    m_Deck.Shuffle();
}

Game::~Game()
{
}
int Game::Menu()
{
    char ch[5];
```

```
    while (1)
    {

        cout << "------------------------\n";
        cout << "Welcome to xzw's 21 points!" << endl;
        cout << " "
             << "Menu" << endl;
        cout << "0:Continue" << endl;
        cout << "1:Single Game\n";
        cout << "2:Multiplayer Game\n";
        cout << "3:Help\n";
        cout << "4:Exit\n";
        cout << "------------------------\n";
        cout << "Your choice:";
        while (cin >> ch)
        {
            if ('0' <= ch[0] && ch[0] <= '4')
                break;
            else
                cout << "Error! Enter again:";
        }
        switch (ch[0])
        {
        case '0':
            return 0;
        case '1':
            return 1;
        case '2':
            return 2;
        case '3':
            return 3;
        case '4':
            return -1;
        }
    }
}

void Game::Save()
{
    ofstream outfile("history.txt", ios::out);
    if (!outfile)
    {
        cerr << "open error!" << endl;
        abort();
    }
    outfile << n << endl;
    for (int i = 0; i < n; i++)
    {
        outfile << m_Players[i].m_Name << " " << m_Players[i].m_Money << endl;
    }
    outfile.close();
}
```

```
void Game::Continue()
{
    ifstream infile("history.txt", ios::in);
    if (!infile)
    {
        cerr << "open error!" << endl;
        abort();
    }
    cout << "loading....." << endl;
    infile >> n;
    for (int i = 0; i < n; i++)
    {
        infile >> m_Players[i].m_Name >> m_Players[i].m_Money;
    }
    infile.close();
}

void Game::Help()
{
    cout << "21 piont is an ancient poker game. The rules of the game are: " << endl;
    cout << "each player tries to make his card score 21 but not more than that. " << endl;
    cout << "Poker cards are valued at their face value. A acts as a score of 1 " << endl;
    cout << "or 11 (a score chosen by the player himself). J. Q and K are both 10  " <<
endl;
    cout << "points. The dealer deals with one to seven players. At the beginning" << endl;
    cout << "of the game, all participants, including the banker, had two cards. " << endl;
    cout << "Players can see all their cards and total points, while the dealer has" <<
endl;
    cout << " a card that is temporarily hidden. Next, each player has the opportunity" <<
endl;
    cout << " to take another card in turn if he wants to. If a players total score " <<
endl;
    cout << "exceeds 21, the player loses. After all players have taken extra cards, " <<
endl;
    cout << "the dealer will show hidden cards. As long as the bankers total score " <<
endl;
    cout << "is less than 17, then he must take another card. If the banker detonates," <<
endl;
    cout << " then all players who have not detonated will win. The player who " << endl;
    cout << "detonated was tied. Otherwise, compare the total score of the remaining " <<
endl;
    cout << "players with the total score of the banker. If the total score of the " <<
endl;
    cout << "player is greater than the total score of the banker, the player wins. " <<
endl;
    cout << "If the total score of the two is the same, the player and the banker draw." <<
endl;
}

void Game::InitUser()
{

    n = 1;
```

```
    cout << "Please enter your name:";
    cin >> m_Players[0].m_Name;
    cout << "How many dollars do you have?";
    cin >> m_Players[0].m_Money;
    cout << "Game begin!" << endl;
}

void Game::InitUsers()
{
    cout << "Welcome to xzw's 21 points!" << endl;
    n = 0;
    while (n < 1 || n > 7)
    {
        cout << "How many people are going to play?(1-7):";
        cin >> n;
    }

    for (int i = 0; i < n; i++)
    {
        cout << "No." << i + 1 << ", please enter your name:";
        cin >> m_Players[i].m_Name;
        cout << "How many dollars do you have?";
        cin >> m_Players[i].m_Money;
    }
    cout << "Game begin!" << endl;
}

void Game::Play()
{
    int i;

    for (i = 0; i < n; i++)
    {
        cout << m_Players[i].m_Name << " ,how many dollars do you want to bet?" << endl;
        int m;
        cin >> m;
        while (m <= 0)
        {
            cout << "Nonlegal! Enter again." << endl;
            cin >> m;
        }
        m_Players[i].m_Bet = m;
    }
    for (int c = 0; c < 2; c++)
    {
        for (i = 0; i < n; i++)
        {
            m_Deck.Deal(m_Players[i]);
        }
        m_Deck.Deal(m_House);
    }

    m_House.FlipFirstCard();
```

```
for (i = 0; i < n; i++)
{
    cout << m_Players[i] << endl;
}
cout << m_House << endl;

for (i = 0; i < n; i++)
{
    m_Deck.AdditionalCards(m_Players[i]);
}

m_House.FlipFirstCard();
cout << endl
     << m_House;

m_Deck.AdditionalCards(m_House);

if (m_House.IsBusted())
{
    for (i = 0; i < n; i++)
    {
        if (!(m_Players[i].IsBusted()))
        {
            m_Players[i].m_Money += 1.5 * m_Players[i].m_Bet;
            m_Players[i].Win();
        }
        else
        {
            m_Players[i].m_Money -= m_Players[i].m_Bet;
            m_Players[i].Lose();
        }
    }
}
else
{
    for (i = 0; i < n; i++)
    {
        if (!(m_Players[i].IsBusted()))
        {
            if (m_Players[i].GetTotal() > m_House.GetTotal())
            {
                m_Players[i].m_Money += 1.5 * m_Players[i].m_Bet;
                m_Players[i].Win();
            }
            else if (m_Players[i].GetTotal() < m_House.GetTotal())
            {
                m_Players[i].m_Money -= m_Players[i].m_Bet;
                m_Players[i].Lose();
            }

            else
            {
```

```
                    m_Players[i].Draw();
                }
            }
            else
            {
                m_Players[i].m_Money -= m_Players[i].m_Bet;
                m_Players[i].Lose();
            }

            if (m_Players[i].m_Money < 0)
                cout << m_Players[i].m_Name << " , you have owed XZW(the house) a lot!" <<
endl;
        }
    }


    for (i = 0; i < n; i++)
    {
        m_Players[i].Clear();
    }
    m_House.Clear();
}
int Game::JudgePlay()
{
    char again;
    cout << "Once again?(Y/N)" << endl;
    cin >> again;
    if (again == 'y' || again == 'Y')
        return 1;
    else
        return 0;
}

ostream &operator<<(ostream &os, const Card &aCard)
{
    const string RANKS[] = {"0", "A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J",
"Q", "K"};
    const string SUITS[] = {"Clubs", "Diamonds", "Hearts", "Spades"};
    if (aCard.m_IsFaceUp)
        os << SUITS[aCard.m_Suit] << RANKS[aCard.m_Rank];
    else
        os << "XX";
    return os;
}
ostream &operator<<(ostream &os, const GenericPlayer &aGenericPlayer)
{
    os << aGenericPlayer.m_Name << ":\t";
    vector<Card *>::const_iterator pCard;
    if (!aGenericPlayer.m_Cards.empty())
    {
        for (pCard = aGenericPlayer.m_Cards.begin();
             pCard != aGenericPlayer.m_Cards.end(); ++pCard)
        {
```

```
            os << *(*pCard) << "\t";
        }
        if (aGenericPlayer.GetTotal() != 0)
            cout << "Total:" << aGenericPlayer.GetTotal() << endl;
    }
    else
    {
        os << "<nothing>";
    }
    return os;
}
```

main.cpp

```
#include "Game.h"
#include <string>
#include <iostream>
using namespace std;

ostream &operator<<(ostream &os, const Card &aCard);
ostream &operator<<(ostream &os, const GenericPlayer &aGenericPlayer);
int main()
{
  Game aGame;
  int ch = 1;
  ch = aGame.Menu();
  while (ch != -1)
  {
    switch (ch)
    {
    case 0:
      aGame.Continue();
      do
      {
        aGame.Play();
      } while (aGame.JudgePlay());
      aGame.Save();
      break;
    case 1:
      aGame.InitUser();
      do
      {
        aGame.Play();
      } while (aGame.JudgePlay());
      aGame.Save();
      break;
    case 2:
      aGame.InitUsers();
      do
      {
        aGame.Play();
      } while (aGame.JudgePlay());
      aGame.Save();
```

```
      break;
    case 3:
      aGame.Help();
      break;
    default:
      exit(0);
    }
    ch = aGame.Menu();
  }

  return 0;
}
```