Student Byoung Mo Lee

Cross Reference for Project 1

You are to fill-in with where located in code

Chapter	Section	Topic	Where Line #"s	Pts	Notes
		Maps (multimap)	Hands.h line 13, 23, 24, 40 Hands.cpp line 14, 134,157		
		Sets	Hands.cpp line 95		
		Lists	Player.h line 14, 26		
		Stacks			
		Queue			
		iterator	Hands.cpp line 95, 134,157		
		algorithm			1. I'll apply to sorting of card object with face and suit 2. Do while error need to be fixed in Dealer.cpp;
		Sum		0	

Class Hierarchy

Go to the graphical class hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

[detail level 1 2]

- C **Card**
- C **Dealer**
- ▼ C **Player**
  - C **Hands**

Card Member List

This is the complete list of members for **Card**, including all inherited members.

Card(int num)	**Card**	
getFace() const	**Card**	inline
getFaceName() const	**Card**	inline
getNumber() const	**Card**	inline
getPict() const	**Card**	inline
getSuit() const	**Card**	inline
getSuitName() const	**Card**	inline
setFace()	**Card**	
setPict()	**Card**	
setSuit()	**Card**	
toString()	**Card**	

Branch: master ▾ Find file Copy path

LeeByoungmo_CIS_17c_47698 / Proj / texasHoldem_17C_Ver1.3 / **Card.h**

libmo No commit message

024972d 1 hour ago

0 contributors

Raw Blame History

42 lines (33 sloc) 834 Bytes

```
/*
 * File:   Card.h
 * Author: Byoung Mo Lee
 * Created on May 16, 2019 23:52 PM
 * Purpose: Texas Holdem
 *
 */

#ifndef CARD_H
#define CARD_H

#include <string>
using namespace std;

class Card{
private:
    int number;
    int face; //2~J(11),Q(12),K(13),A(14)
    int suit;  //0-Spades, 1-Hearts, 2-Diamonds, 3-Clubs
    string faceName;
    string suitName;
    string picture;
public:

    Card(int num);

    void setFace();
    void setSuit();
    void setPict();
    int getFace() const {return this->face;}
    int getSuit() const {return this->suit;}
    string getFaceName() const {return this->faceName;}
    string getSuitName() const { return this->suitName;}
    string getPict() const {return this->picture;}
    int getNumber() const { return this->number;}
    void toString();
};


#endif /* CARD_H */

```

Branch: master ▾ Find file Copy path

LeeByoungmo_CIS_17c_47698 / Proj / texasHoldem_17C_Ver1.3 / Card.cpp

libmo No commit message

024972d 1 hour ago

0 contributors

Raw Blame History

101 lines (91 sloc) 2.98 KB

```
/*
 * File:   Card.cpp
 * Author: Byoung Mo Lee
 * Created on May 17, 2019 00:09 AM
 * Purpose: Texas Holdem
 *
 */

#include <iostream>
#include <iomanip>
#include <string>
#include "Card.h"
using namespace std;

Card::Card(int num){
    if(num>=0&&num<52){
        this->number=num;
        this->setFace();
        this->setSuit();
        this->setPict();
    }
    else{
        this->number=-1;
        this->face=-1;
        this->suit=-1;
        this->faceName="none";
        this->suitName="none";
        this->picture="none";
    }
}

void Card::setFace(){
    if(this->number%13+1==1) {this->face=14;}
    else{this->face=this->number%13+1;}

    switch (this->face) {
            case 2: this->faceName = "two";break;
            case 3: this->faceName = "three";break;
            case 4: this->faceName = "four";break;
            case 5: this->faceName = "five";break;
            case 6: this->faceName = "six";break;
            case 7: this->faceName = "seven";break;
            case 8: this->faceName = "eight";break;
            case 9: this->faceName = "nine";break;
            case 10: this->faceName = "ten";break;
            case 11: this->faceName = "jack";break;
            case 12: this->faceName = "queen";break;
            case 13: this->faceName= "king";break;
            case 14: this->faceName = "ace";break;
            default: this->faceName = "Bad Value";
        }
```

```
}

void Card::setSuit(){
    if(this->number<13){
        this->suit=0;
        this->suitName="spades";
    }
    else if(this->number<26){
        this->suit=1;
        this->suitName="hearts";
    }
    else if(this->number<39){
        this->suit=2;
        this->suitName="diamonds";
    }
    else if(this->number<52){
        this->suit=3;
        this->suitName="clubs";
    }
    else{
        this->suit=-1;
        this->suitName="Bad Value";
    }
}

void Card::setPict(){

    string a[52]={"\U0001F0A1","\U0001F0A2","\U0001F0A3","\U0001F0A4","\U0001F0A5"
            ,"\U0001F0A6","\U0001F0A7","\U0001F0A8","\U0001F0A9","\U0001F0AA","\U0001F0AB"
            ,"\U0001F0AD","\U0001F0AE","\U0001F0B1","\U0001F0B2","\U0001F0B3","\U0001F0B4"
            ,"\U0001F0B5","\U0001F0B6","\U0001F0B7","\U0001F0B8","\U0001F0B9","\U0001F0BA"
            ,"\U0001F0BB","\U0001F0BD","\U0001F0BE","\U0001F0C1","\U0001F0C2","\U0001F0C3"
            ,"\U0001F0C4","\U0001F0C5","\U0001F0C6","\U0001F0C7","\U0001F0C8","\U0001F0C9"
            ,"\U0001F0CA","\U0001F0CB","\U0001F0CD","\U0001F0CE","\U0001F0D1","\U0001F0D2"
            ,"\U0001F0D3","\U0001F0D4","\U0001F0D5","\U0001F0D6","\U0001F0D7","\U0001F0D8"
            ,"\U0001F0D9","\U0001F0DA","\U0001F0DB","\U0001F0DD","\U0001F0DE"};

    for(int i=0;i<52;i++){
        if(this->number==i) this->picture=a[i];
    }
}

void Card::toString(){
    cout << this->Card::getPict() << " ";
    //cout << this->Card::getFace() << " ";
    //cout << this->Card::getSuit() << " ";
    cout << this->Card::getFaceName() << " ";
    cout << this->Card::getSuitName() << endl;
}
```

Player Member List

This is the complete list of members for **Player**, including all inherited members.

addBal(int pot)	**Player**	inline
addMyCards(Card card)	**Player**	
bet(int amount)	**Player**	inline
getActStatus()	**Player**	inline
getBalance() const	**Player**	inline
getMyCards()	**Player**	inline
getName() const	**Player**	inline
getPlayerAct()	**Player**	inline
Player()	**Player**	
putInThePot()	**Player**	inline
resetMyCards()	**Player**	inline
setInThePot()	**Player**	inline
setPlayerAct()	**Player**	inline
setPlayerBal()	**Player**	inline
setPlayerInact()	**Player**	inline
setPlayerName(string name)	**Player**	inline
~Player()	**Player**	inline

Branch: master ▾ Find file Copy path

LeeByoungmo_CIS_17c_47698 / Proj / texasHoldem_17C_Ver1.3 / **Player.h**

libmo No commit message

024972d 1 hour ago

0 contributors

Raw Blame History

52 lines (41 sloc) 1.16 KB

```
/*
 * File:   Player.h
 * Author: Byoung Mo Lee
 * Created on May 17, 2019 10:16 AM
 * Purpose: Player Class for Texas Holdem
 *
 */

#ifndef PLAYER_H
#define PLAYER_H

#include <iostream>
#include <string>
#include <list>
#include "Card.h"


using namespace std;

class Player {
private:
    string name;
    int balance;
    int inThePot;
    bool active;
    list<Card> myCards;

public:
    Player();
    ~Player() {}
    void setPlayerName(string name) {this->name=name;}
    void setPlayerBal() {this->balance=1000;}
    void addMyCards(Card card);
    void setPlayerAct() {this->active=1;}
    string getName() const {return this->name;}
    int getBalance() const {return this->balance;}
    int getPlayerAct() {return this->active;}
    list<Card> getMyCards() {return this->myCards;}
    void bet(int amount) {this->balance -=amount;this->inThePot +=amount;}
    void setPlayerInact() {this->active=0;}
    void setInThePot() {this->inThePot=0;}
    int putInThePot() {return this->inThePot;}
    bool getActStatus() {return this->active;}
    void addBal(int pot) {this->balance += pot;}
    void resetMyCards() {this->myCards.clear();}


};

#endif /* PLAYER_H */

```

Branch: master ▾ Find file Copy path

LeeByoungmo_CIS_17c_47698 / Proj / texasHoldem_17C_Ver1.3 / **Player.cpp**

libmo No commit message

024972d 1 hour ago

0 contributors

Raw Blame History

19 lines (15 sloc) 307 Bytes

```
1  /*
2   * File:   Player.cpp
3   * Author: Byoung Mo Lee
4   * Created on May 27, 2019 10:12 AM
5   * Purpose: Player class for Texas Holdem
6   *
7   */
8  #include <list>
9  #include "Player.h"
10
11 Player::Player(){
12     this->setPlayerBal();
13
14 }
15
16 void Player::addMyCards(Card card){
17         this->myCards.push_back(card);
18     }
```

Hands Member List

This is the complete list of members for **Hands**, including all inherited members.

addBal(int pot)	**Player**	inline
addMyCards(Card card)	**Player**	
bet(int amount)	**Player**	inline
checkStraight()	**Hands**	
getActStatus()	**Player**	inline
getBalance() const	**Player**	inline
getFaces()	**Hands**	inline
getHands()	**Hands**	
getHandsName()	**Hands**	
getMyCards()	**Player**	inline
getName() const	**Player**	inline
getPlayerAct()	**Player**	inline
getSuits()	**Hands**	inline
Hands()	**Hands**	
Hands(const Hands &orig)	**Hands**	
Player()	**Player**	
putInThePot()	**Player**	inline
resetHands()	**Hands**	inline
resetMyCards()	**Player**	inline
setFaces()	**Hands**	
setInThePot()	**Player**	inline
setPlayerAct()	**Player**	inline
setPlayerBal()	**Player**	inline
setPlayerInact()	**Player**	inline
setPlayerName(string name)	**Player**	inline
setSuits()	**Hands**	
~Hands()	**Hands**	inline
~Player()	**Player**	inline

Branch: master ▾ Find file Copy path

LeeByoungmo_CIS_17c_47698 / Proj / texasHoldem_17C_Ver1.3 / **Hands.h**

libmo No commit message

024972d 1 hour ago

0 contributors

Raw Blame History

68 lines (57 sloc) 1.29 KB

```
/*
 * File:   Hands.h
 * Author: Byoung Mo Lee
 * Created on May 18, 2019 20:43 PM
 * Purpose: Hands Class for Texas Holdem
 *
 */


#ifndef HANDS_H
#define HANDS_H

#include <map>
#include "Player.h"

using namespace std;


class Hands : public Player{
private:
    int hands;
    string handsName;
    multimap<int,int, greater<int>> faces;
    map<int,int,greater<int>> suits;
    int isStraight;
    int isFlush;
    int isPair;
    int isStFl;

public:
    //Default constructor
    Hands();
    Hands(const Hands& orig);
    virtual~Hands(){

    }
    void setFaces();
    void setSuits();
    void checkStraight();
    multimap<int,int,greater<int>> getFaces(){return this->faces;}
    map<int,int,greater<int>> getSuits() {return this->suits;}
    int getHands();
    string getHandsName();
    void resetHands(){
        this->resetMyCards();
        this->faces.clear();
        this->suits.clear();
        this->hands=0;
        this->handsName="";
        this->isStraight=0;
        this->isFlush=0;
```

```
        this->isPair=0;
        this->isStFl=0;
    }
};

#endif /* HANDS_H */

// hands=0: Highcard
// hands=1: One Pair
// hands=2: Two Pair
// hands=3: Three of a card
// hands=4: Straight
// hands=5: Flush
// hands=6: Full house
// hands=7: Four of a card
// hands=8: Straight Flush
```

Branch: master ▾ Find file Copy path

LeeByoungmo_CIS_17c_47698 / Proj / texasHoldem_17C_Ver1.3 / **Hands.cpp**

libmo No commit message

024972d 1 hour ago

0 contributors

Raw Blame History

186 lines (169 sloc) 5.27 KB

```
/*
 * File:   Hands.cpp
 * Author: Byoung Mo Lee
 * Created on May 18, 2019 20:43 PM
 * Purpose: Hands Class for Texas Holdem
 *
 */
#include <iostream>
#include <iomanip>
#include <set>
#include <list>
#include <algorithm>
#include <iterator>
#include <map>
#include <string>
#include "Hands.h"

using namespace std;

Hands::Hands():Player(){
    this->Hands::setPlayerAct();
    this->Hands::setInThePot();
}

string Hands::getHandsName(){
    switch (this->Hands::getHands()) {
            case 0: this->handsName = "Highcard";break;
            case 1: this->handsName = "One Pair";break;
            case 2: this->handsName = "Two Pair";break;
            case 3: this->handsName = "Three of a card";break;
            case 4: this->handsName = "Straight";break;
            case 5: this->handsName = "Flush";break;
            case 6: this->handsName = "Full house";break;
            case 7: this->handsName = "Four of a card";break;
            case 8: this->handsName = "Straight Flush";break;
            default: this->handsName = "Bad Value";
        }
    return this->handsName;
}
int Hands::getHands(){
    this->checkStraight();
    this->Hands::setSuits();
    this->Hands::setFaces();

    if(this->isStraight!=0&&this->isFlush!=0){
        this->isStFl=1;
        this->hands=8;
        set<int,greater<int>> stFl;
        for(auto& it: this->Hands::getMyCards()){
            if(it.getSuit()==this->isFlush) {stFl.insert(it.getFace());}
        }
```

```
        if(this->isStraight==15){
            for(int i=14;i>9;i--){
                if(stFl.find(i)==stFl.end()) this->isStFl=0;
            }
        }
        else if(this->isStraight==15){
            for(int i=5;i>1;i--){
                if(stFl.find(i)==stFl.end()) this->isStFl=0;
            }
        }
        else{
            for(int i=this->isStraight;i>this->isStraight-5;i--){
                if(stFl.find(i)==stFl.end()) this->isStFl=0;
            }
        }
        if(this->isStFl==0){
            this->hands=5;
        }
    }
    else if(this->Hands::isPair==7) {this->hands=7;}
    else if(this->isPair==6) {this->hands=6;}
    else if(this->isFlush!=0) {this->hands=5;}
    else if(this->isStraight!=0) {this->hands=4;}
    else if(this->isPair==3) {this->hands=3;}
    else if(this->isPair==2) {this->hands=2;}
    else if(this->isPair==1) {this->hands=1;}
    else {this->hands=0;}
//    cout << endl;
//    for(auto& it: this->Hands::getMyCards()){
//            it.toString();
//    }
//    cout << endl << "Hands=" << this->Hands::getHandsName() << endl;
    return this->hands;
}
void Hands::checkStraight(){
    this->isStraight=0;
    set<int, greater<int>> faceSet;
    int cnt=0;
    for(auto& it: this->Hands::getMyCards()){
        faceSet.insert(it.getFace());
    }
//    cout << endl;

    set<int>::iterator it=faceSet.begin();

    while(it != faceSet.end()&&cnt!=4){
        if(*it-*faceSet.upper_bound(*it)==1) {cnt++;}
        else{cnt=0;}
//        cout << *it << ' ';
        ++it;
    }
    if(cnt==4&&*it+4!=14) {this->isStraight=*it+4;}
    else if(cnt==4&&*it+4==14) {this->isStraight=15;}
    if(*it==14){
        while(it != faceSet.end()&&cnt!=3){
        if(*it-*faceSet.upper_bound(*it)==1) {cnt++;}
        else{cnt=0;}
//        cout << *it << ' ';
        ++it;
        }
    }
    if(cnt==3&&*it+3==5) {this->isStraight=14;}
}


void Hands::setSuits(){
```

```
    this->isFlush=0;
    multiset<int> suitSet;
    for(auto& it: this->Hands::getMyCards()){
        suitSet.insert(it.getSuit());
    }
//    for(int elem:suitSet){
//        cout << elem << ' ';
//    }

    for(int i=0;i<4;i++){
        if(suitSet.count(i)>4) {
            this->suits.insert(pair<int,int>(suitSet.count(i),i));
            this->isFlush=1;
        }
    }
//    cout << endl;
    map<int,int>::iterator pos;
//    for(pos=this->suits.begin();pos!=this->suits.end();++pos){
//        cout << "num: " <<  pos->first << ' '
//             << "suits: " << pos->second << endl;
//    }
//
//    cout << endl;


}
void Hands::setFaces(){
    this->isPair=0;
    multiset<int> faceSet;
    for(auto& it: this->Hands::getMyCards()){
        faceSet.insert(it.getFace());
    }
//    for(int elem:faceSet){
//        cout << elem << ' ';
//    }
    for(int i=2;i<15;i++){
        if(faceSet.count(i)>1) this->faces.insert(pair<int,int>(faceSet.count(i),i));
    }
//    cout << endl;
    map<int,int>::iterator pos;
//    for(pos=this->faces.begin();pos!=this->faces.end();++pos){
//        cout << "pairs: " <<  pos->first << ' '
//             << "faces: " << pos->second << endl;
//    }
//    cout << endl;
//    cout << endl << "faces.empty? " << faces.empty() << endl;
//    cout << endl << "faces.size()? " << faces.size() << endl;

    if(this->faces.empty()) {this->isPair=0;}
    else if(this->faces.size()==1){
        pos=this->faces.begin();
//        cout << "pos->first: " << pos->first << endl;
          if(pos->first==2) this->isPair=1;
          else if(pos->first==3) this->isPair=3;
          else if(pos->first==4) this->isPair=7;

    }
    else if(this->faces.size()>1){
        pos=this->faces.begin();

//        cout << "pos2->first: " << pos->first << endl;
        if(pos->first==2) this->isPair=2;
        else if(pos->first==3) this->isPair=6;
    }


```

```
184 //    cout << "isPair: " << this->isPair << endl;
185
186 }
```

Dealer Member List

This is the complete list of members for **Dealer**, including all inherited members.

bettingPrompt1()	**Dealer**
bettingPrompt2(int num, int amount)	**Dealer**
bettingPrompt3(int num, int amount)	**Dealer**
calBal()	**Dealer**
Dealer(int num)	**Dealer**
decideWinner()	**Dealer**
displayPlayersInfo(int num)	**Dealer**
flop()	**Dealer**
getBigBlind()	**Dealer** inline
getDeck()	**Dealer** inline
getNumAct()	**Dealer**
getNumPlayers() const	**Dealer** inline
getPlayers() const	**Dealer** inline
getPotAmount()	**Dealer** inline
getRound() const	**Dealer** inline
getRound()	**Dealer** inline
getSmallBlind()	**Dealer** inline
nextRound()	**Dealer** inline
preflop()	**Dealer**
resetGame()	**Dealer**
resetPot()	**Dealer** inline
river()	**Dealer**
setBlind()	**Dealer**
setCards()	**Dealer** inline
setIniCont(int num)	**Dealer** inline
setPlayers(int num)	**Dealer**
setRound()	**Dealer** inline
shuffle()	**Dealer**
turn()	**Dealer**
~Dealer()	**Dealer** inline

Branch: master ▾ Find file Copy path

LeeByoungmo_CIS_17c_47698 / Proj / texasHoldem_17C_Ver1.3 / **Dealer.h**

libmo No commit message

024972d 1 hour ago

0 contributors

Raw Blame History

75 lines (65 sloc) 1.81 KB

```
/*
 * File:   Dealer.h
 * Author: Byoung Mo Lee
 * Created on May 17, 2019 09:08 AM
 * Purpose: Texas Holdem
 *
 */
#ifndef DEALER_H
#define DEALER_H

#include <string>
#include "Hands.h"


using namespace std;

class Dealer{
private:
    int potAmount;
    int initialContributeAmount;
    int nRound;   //static
    int numPlayers;
    int bigBlind;
    int smallBlind;
    Hands* players;
    Card* deck[52];
    void DeckOfCards();


public:
    //class NegativeNumber {};
    Dealer(int num);
    ~Dealer() {}
    void shuffle();
    void setPlayers(int num);
    void setCards() {this->DeckOfCards();}
    void setBlind();
    int getBigBlind(){return this->bigBlind;}
    int getSmallBlind(){return this->smallBlind;}
    //void setMyCards(Card* mc) {myCards=mc;}
    void setRound() {this->nRound=0;}
    //void initiateRound();
    //int decideWinner();
    //void completeRound();
    void preflop();
    void flop();
    void turn();
    void river();
    //int getPotAmount() const {return potAmount;}
    //int getInitContAmount() const {return initialContributeAmount;}
    int getRound() const {return this->nRound;}
```

```
    Hands* getPlayers() const {return players;}
    Card** getDeck() {return this->deck;}
    int getNumPlayers() const {return numPlayers;}
    //void displayCards(Card* cards);
    void displayPlayersInfo(int num);
    int bettingPrompt1();
    void setIniCont(int num) {this->initialContributeAmount=num;}
    int bettingPrompt2(int num, int amount);
    int bettingPrompt3(int num, int amount);
    int getPotAmount() {return this->potAmount;}
    int getNumAct();
    int decideWinner();
    void calBal();
    void resetPot(){this->potAmount=0;}
    int resetGame();
    void nextRound() {this->nRound++;}
    int getRound() {return this->nRound;}
};

//int Dealer::round=0;

#endif /* DEALER_H */

```

Branch: master ▾ Find file Copy path

LeeByoungmo_CIS_17c_47698 / Proj / texasHoldem_17C_Ver1.3 / Dealer.cpp

libmo No commit message

024972d 1 hour ago

0 contributors

Raw Blame History

423 lines (354 sloc) 12.4 KB

```
/*
 * File:   Dealer.cpp
 * Author: Byoung Mo Lee
 * Created on May 17, 2019 00:19 AM
 * Purpose: Texas Holdem
 *
 */
#include <string>
#include <iostream>
#include <cstdlib>
#include "Dealer.h"

using namespace std;
Dealer::Dealer(int num){
    int menu=1;
    this->setPlayers(num);
    this->DeckOfCards();
    do{
        this->setBlind();
        this->setIniCont(50);
        this->resetPot();

        this->shuffle();
        this->preflop();

        int amount=bettingPrompt1();
        cout << "bettingPrompt1 is Done" << endl;
    //    for(int i=0;i<this->numPlayers;i++){
    //        cout << this->players[i].getBalance() << endl;
    //    }
    //    cout << this->potAmount << endl;


        int cnt=1;
        int nP=this->numPlayers;
        int x=1;

        do{

            amount=this->bettingPrompt2((cnt+this->smallBlind)%nP,amount);
            if((cnt+this->bigBlind)%nP==this->bigBlind) x=0;
            if(this->Dealer::getNumAct()==1) {
                cout << "find the winner and terminate the round" << endl;
                x=0;
            }
            cnt++;
        }while(amount-this->players[(cnt+this->smallBlind)%nP].putInThePot()||x);
        cout << "bettingPrompt2 is Done" << endl;

        this->flop();
        cnt=0;
```

```
52          x=1;
53          while((amount-this->players[(cnt+this->bigBlind)%nP].putInThePot())||x){
54              int aP=this->Dealer::getNumAct();
55              amount=this->bettingPrompt3((cnt+this->bigBlind)%nP,amount);
56              cnt++;
57              if((cnt+this->bigBlind)%nP==this->bigBlind) x=0;
58              if(this->Dealer::getNumAct()==1) {
59                  cout << "find the winner" << endl;
60                  x=0;
61              }
62
63          }
64          cout << "bettingPrompt3 is Done" << endl;
65
66          this->turn();
67
68          cnt=0;
69          x=1;
70          while((amount-this->players[(cnt+this->bigBlind)%nP].putInThePot())||x){
71              int aP=this->Dealer::getNumAct();
72      //        cout << "ap=" << aP <<endl;
73              amount=this->bettingPrompt3((cnt+this->bigBlind)%nP,amount);
74              cnt++;
75              if((cnt+this->bigBlind)%nP==this->bigBlind) x=0;
76              if(this->Dealer::getNumAct()==1) {
77                  cout << "find the winner" << endl;
78                  x=0;
79              }
80
81          }
82          cout << "bettingPrompt3 is Done" << endl;
83
84          this->river();
85          cnt=0;
86          x=1;
87          while((amount-this->players[(cnt+this->bigBlind)%nP].putInThePot())||x){
88              int aP=this->Dealer::getNumAct();
89              amount=this->bettingPrompt3((cnt+this->bigBlind)%nP,amount);
90              cnt++;
91              if((cnt+this->bigBlind)%nP==this->bigBlind) x=0;
92              if(this->Dealer::getNumAct()==1) {
93                  cout << "find the winner" << endl;
94                  x=0;
95              }
96
97          }
98          cout << "bettingPrompt3 is Done" << endl;
99
100
101
102         for(int i=0;i<this->numPlayers;i++){
103             int order=(i+this->bigBlind)%this->numPlayers;
104             cout << "Player" << order << ": " << this->players[order].getHandsName() << endl;
105
106     //        this->players[order].checkStraight();
107     //        this->players[order].setFaces();
108     //        this->players[order].setSuits();
109     //        this->Dealer::displayPlayersInfo(order);
110         }
111         this->calBal();
112         for(int i=0;i<this->numPlayers;i++){
113             int order=(i+this->bigBlind)%this->numPlayers;
114             this->Dealer::displayPlayersInfo(order);
115         }
116
117         menu=this->resetGame();
```

```
118     }while(menu!=0);
119 }
120 
121 
122 void Dealer::setPlayers(int num){
123     this->numPlayers=num;
124     this->players=new Hands[this->numPlayers];
125 
126     for(int i=0;i<this->numPlayers;i++) {
127               this->players[i].setPlayerName("player#" + to_string(i));
128             }
129 }
130 
131 void Dealer::setBlind(){
132     this->bigBlind=this->getRound()%(this->getNumPlayers()-1);
133     this->smallBlind=this->bigBlind+1;
134 }
135 
136 
137 void Dealer::displayPlayersInfo(int num){
138     cout << this->Dealer::players[num].getName() << endl;
139     if(num==this->Dealer::getBigBlind()) {cout << "BB" << endl;}
140     else if (num==this->Dealer::getSmallBlind()) {cout << "SB" << endl;}
141     cout << "Balance: $" <<this->Dealer::players[num].getBalance() << endl << endl;
142     for(auto& it:this->players[num].getMyCards()){
143         it.toString();
144     }
145     cout << endl;
146 }
147 
148 void Dealer::DeckOfCards() {
149     for(int i=0;i<52;i++){
150         this->deck[i]=new Card(i);
151         //cards[i]->toString();
152     }
153 }
154 
155 
156 
157 void Dealer::shuffle(){
158     // Initialize seed randomly
159     srand(time(0));
160 
161     for (int i=0; i<52 ;i++)
162     {
163         // Random for remaining positions.
164         int r = i + (rand() % (52 -i));
165 
166         swap(this->deck[i], this->deck[r]);
167     }
168 
169 }
170 
171 void Dealer::preflop(){
172     int nCards=2;
173     int order;
174     int nP=this->numPlayers;
175     int bB=this->bigBlind;
176     for(int i=0;i<nCards;i++){
177         for(int j=0;j<nP;j++){
178             order=(j+bB)%nP;
179             this->players[order].addMyCards(*this->deck[j+nP*i]);
180 
181 //            this->deck[j+i+nP*i]->toString();
182 //            this->Dealer::displayPlayersInfo(order);
183 
```

```
        }
    }
}

void Dealer::flop(){
    int nCards=3;
    int nP=this->numPlayers;
    int bB=this->bigBlind;
    for(int i=nP*2+1;i<nP*2+1+nCards;i++){
//        this->deck[i]->toString();
        for(int j=0;j<nP;j++){
            int order=(j+bB)%nP;
            this->players[order].addMyCards(*this->deck[i]);
        }
    }
}

void Dealer::turn(){
    int nCards=1;
    int nP=this->numPlayers;
    int bB=this->bigBlind;
    for(int i=nP*2+1+3+1;i<nP*2+1+3+1+nCards;i++){
//        this->deck[i]->toString();
        for(int j=0;j<nP;j++){
            int order=(j+bB)%nP;
            this->players[order].addMyCards(*this->deck[i]);
        }
    }
}

void Dealer::river(){
    int nCards=1;
    int nP=this->numPlayers;
    int bB=this->bigBlind;
    for(int i=nP*2+1+3+1+1+1;i<nP*2+1+3+1+1+1+nCards;i++){
//        this->deck[i]->toString();
        for(int j=0;j<nP;j++){
            int order=(j+bB)%nP;
            this->players[order].addMyCards(*this->deck[i]);
        }
    }
}

int Dealer::bettingPrompt1(){
    int input;
    int raise;
    int diff;
    int nP=this->numPlayers;
    int amount;
    this->players[bigBlind].bet(this->initialContributeAmount);
    this->potAmount+=this->initialContributeAmount;
    cout << "Player" << this->bigBlind << ": $" << this->initialContributeAmount <<" into the Pot"<<endl;
    cout << "Pot: $" << this->getPotAmount() << endl;
    amount=this->players[bigBlind].putInThePot();

    this->players[smallBlind].bet(this->initialContributeAmount/2);
    this->potAmount+=this->initialContributeAmount/2;
    diff=amount-this->initialContributeAmount/2;


    cout << "Player" << this->smallBlind << ": $" << this->initialContributeAmount/2<<" into the Pot" <<endl;
    cout << "Pot: $" << this->getPotAmount() << endl;


    do{
    cout << "To call you have to put $" << diff << endl;
```

```
    cout << "Player" << this->smallBlind <<": Call - 1, Raise - 2, Fold -3" << endl;
    cin >> input;
    if(input==1) {
            this->players[smallBlind].bet(diff);
            this->potAmount+=diff;
            cout << "Player#" << smallBlind << ": $" << diff << endl;
            cout << "Pot: $" << this->getPotAmount() << endl;

        }
        else if(input==2){
            cout << "Amount: " ;
            do{
                cin >> raise;
                if(raise<=diff) cout << "Amount should be greater than call Amount\n";
            }while(raise<=diff);
            this->players[smallBlind].bet(raise+diff);
            this->potAmount+=(raise+diff);
            cout << "Player#" << smallBlind << ": $" << raise+diff <<" into the Pot" << endl;
            cout << "Pot: $" << this->getPotAmount() << endl;
            amount = this->players[smallBlind].putInThePot();
        }
        else if(input==3){
            this->players[smallBlind].setPlayerInact();
            cout << "Player#" << smallBlind << ": folded" << endl;
            cout << "Pot: $" << this->getPotAmount() << endl;
        }
        else {cout << "wrong input" << endl;}
    }while(input<1||input>3);

    return amount;
}

int Dealer::bettingPrompt2(int num, int amount){
    int input;
    do{
        if(this->players[num].getActStatus()){
            int nP=this->numPlayers;

            int raise;
            int diff=amount - this->players[num].putInThePot();
            cout << "To call you have to put $" << diff << endl;
            cout << "Player" << num <<": call - 1, Raise - 2, Fold -3" << endl;
            cin >> input;
            if(input==1) {
                this->players[num].bet(diff);
                this->potAmount+=(diff);
                cout << "Player#" << num << ": $" << diff << endl;
                cout << "Pot: $" << this->getPotAmount() << endl;
            }
            else if(input==2){
                cout << "Amount: " ;
                do{
                    cin >> raise;
                    if(raise<=diff) cout << "Amount should be greater than call Amount\n";
                }while(raise<=diff);

                this->players[num].bet(raise+diff);
                this->potAmount+=(raise+diff);
                cout << "Player#" << num << ": $" << raise+diff << endl;
                cout << "Pot: $" << this->getPotAmount() << endl;
                amount=this->players[num].putInThePot();
            }
            else if(input==3){
                this->players[num].setPlayerInact();
                cout << "Player#" << num << ": folded" << endl;
                cout << "Pot: $" << this->getPotAmount() << endl;
```

```
            }
            else {cout << "wrong input" << endl;}
        }
    }while(input<1||input>3);

    return amount;
}

int Dealer::bettingPrompt3(int num, int amount){
    int nP=this->numPlayers;
    int input;
    do{
        if(this->players[num].getActStatus()){
            int raise;
            int diff=amount - this->players[num].putInThePot();
            if(!diff){
                cout << "You can check" << endl;
                cout << "Player" << num <<": check - 1, Raise - 2, Fold -3" << endl;

            }
            else{
                cout << "To call you have to put $" << diff << endl;
                cout << "Player" << num <<": call - 1, Raise - 2, Fold -3" << endl;

            }

            cin >> input;
            if(input==1) {

                this->players[num].bet(diff);
                this->potAmount+=(diff);
                cout << "Player#" << num << ": $" << diff << endl;
                cout << "Pot: $" << this->getPotAmount() << endl;
            }
            else if(input==2){
                cout << "Amount: " ;
                do{
                    cin >> raise;
                    if(raise<=diff) cout << "Amount should be greater than call Amount\n";
                }while(raise<=diff);

                this->players[num].bet(raise+diff);
                this->potAmount+=(raise+diff);
                cout << "Player#" << num << ": $" << raise+diff << endl;
                cout << "Pot: $" << this->getPotAmount() << endl;
                amount=this->players[num].putInThePot();
            }
            else if(input==3){
                this->players[num].setPlayerInact();
                cout << "Player#" << num << ": folded" << endl;
                cout << "Pot: $" << this->getPotAmount() << endl;
            }
            else {cout << "wrong input" << endl;}
        }
    }while(input<1||input>3);

    return amount;
}

int Dealer::getNumAct(){
    int num=0;
    for(int i=0;i<this->numPlayers;i++){
        if(this->players[i].getActStatus()) num++;
    }

    return num;
```

```
}

int Dealer::decideWinner(){
    int max=0;
    int winner=0;
    int nP=this->getNumPlayers();

    for(int i=0;i<nP;i++){
        if(this->players[i].getActStatus()==1) {
            if(this->players[i].getHands()>max) {
                max=this->players[i].getHands();
                winner=i;
            }
        }
    }
//    cout << "Max=" << max << endl;
    return winner;

}

void Dealer::calBal(){
    int winner=this->decideWinner();
    cout << "Player" << winner << " won $" << this->getPotAmount() << "!" << endl;
    this->players[winner].addBal(this->getPotAmount());
    this->Dealer::resetPot();
}

int Dealer::resetGame(){
    int input;
    this->nextRound();

    for(int i=0;i<this->getNumPlayers();i++){

        this->players[i].resetHands();
    }


    cout << "if you want to exit press 0" << endl;
    cin >> input;

    return input;
}
```