Drew Sweeney

Dr. Hu

Homework 2

Reflection:

This assignment was challenging but I really enjoyed it. Comparing this assignment to last semester 241 I feel like I can visualize the code a lot better, and it makes it a lot easier and a lot more enjoyable to do the assignment for me. Connecting the parts together and having an output that I can see and understand helps me a lot. I think I nailed the extra credit as well by inserting a human player to play alongside the AI's.

Card

```
package homework2;

import homework2.Interface.TheCard;

public class Card implements TheCard
{
	private int number;
	private int rank;
	private int suit;
	private int pairNumber;

	public Card(int S)
	{
		number = S;
		rank = S/4;
		suit = S%4;
		pairNumber = setPairNumber();
	}
	public Card(int S, int T)
	{
		number = (S-1)*4+T-1;
		rank = S-1;
		suit = T-1;
		pairNumber = setPairNumber();
	}
	public Card()
	{
		number = 1;
		rank = 0;
		suit = 0;
	}
	public int getNumber()
	{
		return number;
	}
```

```
public int getRank()
{
	return rank;
}
public int getSuit()
{
	return suit;
}
public int getPairNumber()
{
	return pairNumber;
}
public String getRankCard()
{
	String rankfr = new String();
	switch (rank) {
	case 0:  rankfr = "ACE";
	break;
	case 1:  rankfr = "TWO";
	break;
	case 2:  rankfr = "THREE";
	break;
	case 3:  rankfr = "FOUR";
	break;
	case 4:  rankfr = "FIVE";
	break;
	case 5:  rankfr = "SIX";
	break;
	case 6:  rankfr = "SEVEN";
	break;
	case 7:  rankfr = "EIGHT";
	break;
	case 8:  rankfr = "NINE";
	break;
	case 9:  rankfr = "TEN";
	break;
	case 10: rankfr = "JACK";
	break;
	case 11: rankfr = "QUEEN";
	break;
	case 12: rankfr = "KING";
	break;
	default: rankfr = "Invalid rank";
	break;
	}
	return rankfr;
}
public String getSuitCard()
{
	String suitfr = new String();
	switch (suit) {
	case 0:  suitfr = "HEARTS";
	break;
	case 1:  suitfr = "DIAMONDS";
	break;
```

```
                case 2:  suitfr = "CLUBS";
                break;
                case 3:  suitfr = "SPADES";
                break;
                default: suitfr = "Invalid suit";
                break;
                }
                return suitfr;
        }
        public String getCard()
        {
                return getRankCard()+" of "+getSuitCard();
        }
        public void printCard()
        {
                System.out.println(getRankCard()+" of "+getSuitCard());
        }
        public int setPairNumber()
        {
                if (suit<2)
                {
                        pairNumber=suit*13+rank;
                }
                else if (suit==2)
                {
                        pairNumber=3*13+12-rank;
                }
                else if (suit==3)
                {
                        pairNumber=2*13+12-rank;
                }
                return pairNumber;
        }
        @Override
        public boolean isPair(Card two)
        {
                if (this.pairNumber + two.getPairNumber() == 51)
                {
                        return true;
                }
                else
                {
                        return false;
                }
        }
}
```

Hand

```
package homework2;

import java.util.*;

import homework2.Interface.TheHand;
```

```
public class Hand extends Deck implements TheHand
{
	boolean human;
	String nameCPU;
	boolean firstPlayer;

	public Hand(int S, Deck pick, boolean B, Card removed)
	{
		super(0);
		human=B;

		String names[]={"Drew", "Dr. Hu", "Noah", "Hayden", "Josh", "Bobby",
"Peaches", "Fruit", "Mike", "Ethan"};

		Random generator = new Random();
		int random = generator.nextInt(names.length);
		nameCPU=names[random];
		Card newCard;

		firstPlayer = false;

		for (int i = 0; i < S; i++)
		{
			newCard=takeCard(pick,0);
			if ( ( (removed.getNumber() ) == 51) )
			{
				if ( (newCard.getNumber()) == 25)
				{
					firstPlayer = true;
				}
			}
			else if ( (newCard.getNumber() ) == 51)
			{
				firstPlayer = true;
			}
		}

		if(B)
		{
			showCards();
		}
	}
	public String getNameCPU()
	{
		return nameCPU;
	}

	public boolean getFirstPlayer()
	{
		return firstPlayer;
	}

	public Card takeRandomCard(Deck two)
	{
```

```
                Random generator = new Random();
                int numTaken = generator.nextInt(two.getNumber() );

                return addCardTop(two.drawCard(numTaken) );
        }
}
```

Deck

```
package homework2;

import java.util.*;

import homework2.Interface.TheDeck;

public class Deck implements TheDeck
{
        int number;
        ArrayList<Integer> numTaken;
        ArrayList<Integer> numNotTaken;

        public Deck(int S)
        {
                number = 0;
                numTaken= new ArrayList<Integer>();
                numNotTaken= new ArrayList<Integer>();

                for (int i = 0; i < 52; i++)
                {
                        numNotTaken.add(i);
                }

                Collections.shuffle(numNotTaken);

                for (int i = 0; i < S; i++)
                {
                        addCardTop();
                }
        }
        public Deck()
        {
                number = 52;
                numTaken= new ArrayList<Integer>();
                numNotTaken= new ArrayList<Integer>();

                for (int i = 0; i < number; i++)
                {
                        numTaken.add(i);
                }

                Collections.shuffle(numTaken);
        }

        public ArrayList<Integer> getNumTaken()
```

```
{
	return numTaken;
}

public ArrayList<Integer> getNumNotTaken()
{
	return numNotTaken;
}

public int getNumber()
{
	return number;
}

public Card drawCard()
{
	Random generator = new Random();
	int notDrew = generator.nextInt(numTaken.size());
	Card takenCard = new Card(numTaken.get(notDrew));
	Integer Ndrew = new Integer(notDrew);
	numNotTaken.add(Ndrew);
	shuffleAntiCards();
	numTaken.remove(notDrew);
	number--;

	return takenCard;
}

public Card drawCard(int position)
{
	Card takenCard = new Card(numTaken.get(position));
	Integer Ndrew = new Integer(position);
	numNotTaken.add(Ndrew);
	shuffleAntiCards();
	numTaken.remove(position);
	number--;
	return takenCard;
}

public Card addCardTop()
{
	Card ToBeAdded = null;
	if (number == 52)
	{
		System.out.println("Impossible");
	}
	else
	{
		numTaken.add(numNotTaken.get(0) );
		int number = (numNotTaken.get(0) ).intValue();
		ToBeAdded = new Card(number);
		numNotTaken.remove(0);
	}

	number++;
```

```
        return ToBeAdded;
}

public Card addCardTop(Card added)
{
        int cardNumber =added.getNumber();

        if (number==52)
        {
                System.out.println("Impossible");
        }
        else
        {
                if (number==0)
                {
                        numTaken.add(cardNumber);
                }
                else
                {
                        numTaken.add(cardNumber);
                }

                for(int i = 0; i < 52 - number; i++)
                {
                        if ((numNotTaken.get(i) ) == cardNumber)
                        {
                                numNotTaken.remove(i);
                                number++;
                                return added;
                        }
                }
        }
        number++;
        return added;
}

public Card addCard()
{
        Card ToBeAdded = null;
        if (number==52) System.out.println("Impossible");
        else
        {
                Random generator = new Random();
                int input = generator.nextInt(numTaken.size());
                numTaken.add(input,numNotTaken.get(0));
                int number = (numNotTaken.get(0)).intValue();
                ToBeAdded=new Card(number);
                numNotTaken.remove(0);
        }
        number++;
        return ToBeAdded;
}

public Card addCard(Card added)
```

```
	{
		int cardNumber = added.getNumber();
		if (number == 52)
		{
			System.out.println("Impossible");
		}
		else
		{
			Random generator = new Random();
			if (number == 0)
			{
				numTaken.add(cardNumber);
			}
			else
			{
				int input = generator.nextInt(number);
				numTaken.add(input, cardNumber);
			}
			for(int i = 0; i < 52 - number; i++)
			{
				if ( (numNotTaken.get(i) ) == cardNumber)
				{
					numNotTaken.remove(i);
					number++;
					return added;
				}
			}
		}

		number++;

		return added;
	}

	public void showCards()
	{
		int numberS;
		String zero = new String("");
		for (int i = 0; i < number; i++)
		{
			numberS = i + 1;
			if (numberS < 10)
			{
				zero="0";
			}
			else
			{
				zero="";
			}
			Card cardinhand = new Card(numTaken.get(i) );
			System.out.println("Card number: " + zero + numberS + " : " +
cardinhand.getRankCard() + " of "+ cardinhand.getSuitCard() );
		}
	}
```

```
	public void printCard(int position)
	{
		Card card = new Card(numTaken.get(position));
		card.printCard();
	}

	public void shuffleCards()
	{
		Collections.shuffle(numTaken);
	}

	public void shuffleAntiCards()
	{
		Collections.shuffle(numNotTaken);
	}

	public int calculatePairNumber(int S)
	{
		int rank = S / 4;
		int suit = S % 4;
		int pairNumber = 0;
		if (suit < 2)
		{
			pairNumber= suit * 13 + rank;
		}
		else if (suit == 2)
		{
			pairNumber = 3 * 13 + 12 - rank;
		}
		else if (suit == 3)
		{
			pairNumber = 2 * 13 + 12 - rank;
		}

		return pairNumber;
	}

	public boolean isPair(int i, int j)
	{
		if (calculatePairNumber(numTaken.get(i) ) +
calculatePairNumber(numTaken.get(j) ) == 51)
		{
			return true;
		}
		else
		{
			return false;
		}
	}

	public boolean isPair(Card one, Card two)
	{
		if (one.getPairNumber() + two.getPairNumber() == 51)
		{
			return true;
```

```
        }
        else
        {
                return false;
        }
}

public boolean isDouble()
{
        boolean ok=false;
        for (int i = 0; i < number - 1 && !ok; i++)
        {
                if (isPair(i,number - 1) )
                {
                        removePairs(i, number - 1);
                        ok = true;
                }
        }

        return ok;
}

public Card lookCard(Deck two, int cardPosition)
{
        Card looked = new Card( (two.getNumTaken() ).get(cardPosition) );
        return looked;
}

public Card takeCard(Deck two, int cardPosition)
{
        return addCardTop(two.drawCard(cardPosition));
}

public void removePairs(int i, int j)
{
        System.out.print("  ");
        printCard(i);

        System.out.print("+ ");
        printCard(j);

        System.out.print("\n");
        drawCard(i);
        drawCard(j - 1);
}

public int removePairs()
{
        int drawndoubles=0;
        for(int i = 0, j; i < number ;i++)
        {
                for(j = i + 1; j < number; j++)
                {
                        if (isPair(i, j) )
                        {
```

```
                                removePairs(i,j);
                                drawndoubles++;
                                j=i;
                        }
                }
                if (j != number)
                {
                        i--;
                }
        }

        return drawndoubles;
    }
}
```

Main

```
package homework2;

public class Main
{
    public static void main(String[] args)
    {
        // TODO Auto-generated method stub
        Game game = new Game();
    }
}
```

Game

e homework2;

import java.util.*;

import homework2.Interface.TheGame;

public class Game implements TheGame

{

Scanner sc;

int numOfPlayers;

int playersPlaying;

```
        Deck deck;
        String answer;
        ArrayList <Hand> hands;
        Card removed;
        int turn;
        ArrayList <Integer> turnsPlayed;

        public Game()
        {

                sc = new Scanner(System.in);
                deck = new Deck(52);
                Deck used = new Deck(0);
                System.out.println("Give a number between 1 and 52: ");

                int removedCard =  sc.nextInt();

                removed = deck.lookCard(deck, removedCard - 1);
                used.takeCard(deck, removedCard - 1);


                System.out.println("Give the number of players including yourself");

                numOfPlayers =  sc.nextInt();
                playersPlaying = numOfPlayers;
                sc.nextLine();
                askShuffle();

                int cardsperplayer = 51 / numOfPlayers;
```

```
int leftcards = 51 - numOfPlayers*cardsperplayer;
hands = new ArrayList <Hand> ();
System.out.println("Your starting hand is: ");
hands.add(new Hand(cardsperplayer, deck, true, removed));
turnsPlayed = new ArrayList <Integer>();
turnsPlayed.add(0);

for (int i = 0; i < numOfPlayers - 1; i++)
{
    turnsPlayed.add(0);
    hands.add(new Hand(cardsperplayer,deck,false,removed));
    if (leftcards>0)
    {
        (hands.get(i)).takeCard(deck,0);
        leftcards--;
    }
}

System.out.println("");

System.out.println("Your pairs are: ");
(hands.get(0)).removePairs();
for (int i = 1; i< numOfPlayers; i++)
{
    System.out.println( (hands.get(i) ).getNameCPU() + "s pairs are: ");
    (hands.get(i) ).removePairs();
}
turn=1;
boolean continuegame;
```

```
                playFirstTurn();
                turn++;
                while (continuegame=playTurn())
                {
                        checkHands();
                        System.out.println("Game starts on turn " + turn);
                        if(continuegame) turn++;
                }
                System.out.println("Game finished on turn " + turn);
                stats();
        }

        public void askToPickCard(int playernumber)
        {
                int playerNumber = playernumber-1;
                int max = (hands.get(playerNumber) ).getNumber();
                int pos = 0;
                do
                {
                        if (pos > max) System.out.print("\nError");
                        System.out.println("You remove a card from the next player, " +
(hands.get(playerNumber) ).getNameCPU() + " (" + (playerNumber+1) + "). Give a position (1-" + max
+").");

                        pos =  sc.nextInt();
                        sc.nextLine();

                }while(pos > max);
                pos--;
```

```
                Card newcard = (hands.get(0) ).takeCard(hands.get(playerNumber),pos);
                System.out.print("You remove a card from the next player,  " +
(hands.get(playerNumber) ).getNameCPU() + " (" + (playerNumber + 1) + ") : ");
                newcard.printCard();
                if(!( (hands.get(0) ).isDouble() ) )
                {
                        System.out.println("Too bad... No new pair was created");
                }

        }

        public void askShuffle()
        {
                System.out.println("Do you want to shuffle your hand [Y/N]");

                boolean ok = false;
                do
                {
                        answer =  sc.nextLine();
                        ok = (answer.equals("Y") ) || (answer.equals("y") ) || (answer.equals("N") ) ||
(answer.equals("n") );
                        if (!ok)
                        {
                                System.out.print("\nError");
                        }

                }while (!ok);

                if ( (answer.equals("N") ) || (answer.equals("n") ) )
```

```
            {
                System.out.println("");
            }
        }

        public void shufflePlayer()
        {
            if ( (answer.equals("Y") ) || (answer.equals("y") ) )
            {
                (hands.get(0) ).shuffleCards();

                System.out.println("Shuffled hand: ");

                (hands.get(0) ).showCards();
            }
        }

        public boolean playTurnUser()
        {
            if ( ( (hands.get(0)).getNumber() ) == 0)
            {
                return false;
            }
            boolean ok = false;

            for (int i=numOfPlayers;i>1&&!ok;i--)
            {
                ok = ( ( (hands.get(i-1) ).getNumber() ) !=0);
                if (ok)
```

```
                    {
                        askToPickCard(i);
                    }
                }

                if ( ( (hands.get(0) ).getNumber() ) == 0)
                {
                    return false;
                }
                else
                {
                    System.out.println("Your hand: ");
                    (hands.get(0) ).showCards();
                    System.out.println("");

                    shufflePlayer();
                    System.out.println("");

                    return true;
                }
            }

            public boolean playTurnCPU(int playernumber)
            {
                int playerNumber = playernumber - 1;

                if ( ( (hands.get(playerNumber) ).getNumber() ) ==0)
                {
                    return false;
```

```
        }
        Card taken = null;
        boolean ok = false;

        for (int i = playerNumber; i > 0 && !ok; i--)
        {
            ok = (((hands.get(i-1)).getNumber())!=0);
            if (ok)
            {
                taken = (hands.get(playerNumber) ).takeRandomCard(hands.get(i - 1)
);

                if (playerNumber == 1)
                {
                    System.out.print( (hands.get(playerNumber) ).getNameCPU()
+ " (" + (playerNumber + 1) +") Take your card: ");
                }
                else
                {
                    System.out.print( (hands.get(playerNumber) ).getNameCPU()
+ " (" + (playerNumber + 1) + ") take card of "+(hands.get(i - 1) ).getNameCPU() + " (" + (i) + ") : ");
                }
                taken.printCard();
                System.out.println("");
                (hands.get(playerNumber) ).isDouble();
            }
        }

        if (taken == null)
        {
```

```
                    for (int i = numOfPlayers; i > playerNumber + 1 && !ok; i--)
                    {
                        ok = (((hands.get(i-1)).getNumber())!=0);
                        if (ok)
                        {
                            taken = (hands.get(playerNumber)).takeRandomCard(hands.get(i-1));
                            if (playerNumber == 1)
                            {
                                System.out.print( (hands.get(playerNumber) ).getNameCPU() + " (" + (playerNumber + 1) + ") Take your card: ");
                            }
                            else
                            {
                                System.out.print( (hands.get(playerNumber) ).getNameCPU()  + " (" + (playerNumber+1) + ") take card of " + (hands.get(i - 1) ).getNameCPU() + " (" + (i) + ") : ");
                            }
                            taken.printCard();

                            System.out.println("");

                            (hands.get(playerNumber) ).isDouble();
                        }
                    }
                }


                if ( ( (hands.get(playerNumber)).getNumber() ) == 0)
                {
                    return false;
```

```
            }
            else
            {
                (hands.get(playerNumber) ).shuffleCards();
                return true;
            }
        }

        public int qualify(int playernumber)
        {
            int playerNumber = playernumber - 1;

            if (playerNumber == 0)
            {
                System.out.println("You no longer have a card! Nice job!");
            }

            turnsPlayed.set(playerNumber,turn);

            playersPlaying--;

            return turn;
        }

        public boolean playFirstTurn()
        {
            if ( (hands.get(0) ).getFirstPlayer() )
            {
                System.out.println("You have the queen...Start!");
```

```
                return true;
            }
            else
            {
                int firstone = - 1;
                for (int i = 1; i < numOfPlayers; i++)
                {
                    if ( (hands.get(i) ).getFirstPlayer() ) firstone = i;
                }

                boolean toqualify=false;

                System.out.println( (hands.get(firstone) ).getNameCPU() +" (" + (firstone + 1) +
") Begin! ");


                for (int i=firstone;i<numOfPlayers;i++)
                {
                    if ( (turnsPlayed.get(i) ) == 0)
                    {
                        toqualify=!playTurnCPU(i+1);
                    }

                if (toqualify && ( (turnsPlayed.get(i) ) == 0) )
                {
                    System.out.println( (hands.get(i) ).getNameCPU() + " (" + (i + 1)
+ ") no more cards");
                    qualify(i + 1);

                    if (playersPlaying==1)
```

```
                                        {
                                                return false;
                                        }
                                }
                        }
                        System.out.println("Number of players is: "+playersPlaying);
                }
                if (playersPlaying==1)
                {
                        return false;
                }
                else
                {
                        return true;
                }
        }

        public boolean playTurn()
        {
                boolean toqualify=false;


                if ( (turnsPlayed.get(0) ) == 0)
                {
                        toqualify =! playTurnUser();
                }
                if (toqualify && ( (turnsPlayed.get(0) ) == 0) )
                {
                        System.out.println("You are good!");
                        qualify(1);
```

```
                    if (playersPlaying==1) return false;
            }

            for (int i = 1; i < numOfPlayers; i++)
            {
                    if ((turnsPlayed.get(i) ) == 0)
                    {
                            toqualify =! playTurnCPU(i + 1);
                    }
                    if (toqualify && ( (turnsPlayed.get(i) ) == 0) )
                    {
                            System.out.println( (hands.get(i) ).getNameCPU() + " (" + (i + 1) + ") no
more cards");
                            qualify(i + 1);
                            if (playersPlaying == 1)
                            {
                                    return false;
                            }
                    }
            }
            System.out.println("Number of players is: " + playersPlaying);
            if (playersPlaying == 1)
            {
                    return false;
            }
            else
            {
                    return true;
            }
```

```
        }

        public void checkHands()
        {
                System.out.println("You have "+(hands.get(0)).getNumber()+" cards");
                for (int i=1;i<numOfPlayers;i++)
                {
                        System.out.println((hands.get(i)).getNameCPU()+" ("+(i+1)+") a :
"+(hands.get(i)).getNumber()+" cards");
                }
        }

        public void stats()
        {
                int nloser=-1;

                if ( (turnsPlayed.get(0) ) != 0)
                {
                        System.out.println("You won after "+turnsPlayed.get(0) + " turns!");
                }

                for (int i=1;i<numOfPlayers;i++)
                {
                        turnsPlayed.add(0);

                        if ( (turnsPlayed.get(i) ) == 0)
                        {
                                nloser = i;
                        }
```

```
                    else
                    {
                        System.out.println( (hands.get(i) ).getNameCPU() + " ("+(i + 1) + ") won after "+turnsPlayed.get(i) + " turns!");
                    }
                }
                if (nloser == -1)
                {
                    System.out.println("After " + turn + " turns you lost!");

                    System.out.println("You had the queen: ");
                    (hands.get(0)).printCard(0);

                    System.out.print("You removed the: ");

                    removed.printCard();
                }
                else
                {
                    System.out.println((hands.get(nloser)).getNameCPU()+" ("+(nloser+1)+") lost after " + turn + " turns!");
                }
            }
}
```

Interface

```
package homework2;

import java.util.ArrayList;

public class Interface
{
```

```
interface TheDeck
{
	public ArrayList<Integer> getNumTaken();

	public ArrayList<Integer> getNumNotTaken();

	public int getNumber();

	public Card drawCard();

	public Card drawCard(int position);

	public Card addCardTop();

	public Card addCardTop(Card added);

	public Card addCard();

	public Card addCard(Card added);

	public void showCards();

	public void printCard(int position);

	public void shuffleCards();

	public void shuffleAntiCards();

	public int calculatePairNumber(int S);

	public boolean isPair(int i, int j);

	public boolean isPair(Card one, Card two);

	public boolean isDouble();

	public Card lookCard(Deck two, int cardPosition);

	public Card takeCard(Deck two, int cardPosition);

	public void removePairs(int i, int j);

	public int removePairs();


}

interface TheHand
{
	public String getNameCPU();

	public boolean getFirstPlayer();

	public Card takeRandomCard(Deck two);
```

```
	}

	interface TheCard
	{
		public int getNumber();

		public int getRank();

		public int getSuit();

		public int getPairNumber();

		public String getRankCard();

		public String getSuitCard();

		public String getCard();

		public void printCard();

		public int setPairNumber();

		public boolean isPair(Card two);
	}

	interface TheGame
	{
		public void askToPickCard(int playernumber);

		public void askShuffle();

		public void shufflePlayer();

		public boolean playTurnUser();

		public boolean playTurnCPU(int playernumber);

		public int qualify(int playernumber);

		public boolean playFirstTurn();

		public boolean playTurn();

		public void checkHands();

		public void stats();
	}
}
```

Output

```
Give a number between 1 and 52:
```

```
2
Give the number of players including yourself
2
Do you want to shuffle your hand [Y/N]
n

Your starting hand is:
Card number: 01 : SEVEN of SPADES
Card number: 02 : ACE of SPADES
Card number: 03 : JACK of HEARTS
Card number: 04 : EIGHT of HEARTS
Card number: 05 : FIVE of CLUBS
Card number: 06 : TEN of DIAMONDS
Card number: 07 : FOUR of SPADES
Card number: 08 : QUEEN of DIAMONDS
Card number: 09 : TEN of HEARTS
Card number: 10 : TEN of SPADES
Card number: 11 : ACE of DIAMONDS
Card number: 12 : QUEEN of SPADES
Card number: 13 : SIX of DIAMONDS
Card number: 14 : KING of SPADES
Card number: 15 : NINE of DIAMONDS
Card number: 16 : NINE of CLUBS
Card number: 17 : NINE of SPADES
Card number: 18 : KING of DIAMONDS
Card number: 19 : JACK of DIAMONDS
Card number: 20 : SIX of HEARTS
Card number: 21 : TEN of CLUBS
Card number: 22 : EIGHT of DIAMONDS
Card number: 23 : FIVE of DIAMONDS
Card number: 24 : SIX of SPADES
Card number: 25 : EIGHT of CLUBS

Your pairs are:
  ACE of SPADES
+ ACE of DIAMONDS

  EIGHT of HEARTS
+ EIGHT of CLUBS

  TEN of DIAMONDS
+ TEN of SPADES

  QUEEN of DIAMONDS
+ QUEEN of SPADES

  TEN of HEARTS
+ TEN of CLUBS

  SIX of DIAMONDS
+ SIX of SPADES

  KING of SPADES
+ KING of DIAMONDS
```

```
  NINE of DIAMONDS
+ NINE of SPADES

  SIX of HEARTS
+ SIX of CLUBS

Fruits pairs are:
  TWO of DIAMONDS
+ TWO of SPADES

  THREE of DIAMONDS
+ THREE of SPADES

  TWO of CLUBS
+ TWO of HEARTS

  ACE of HEARTS
+ ACE of CLUBS

  KING of HEARTS
+ KING of CLUBS

  FOUR of HEARTS
+ FOUR of CLUBS

  THREE of CLUBS
+ THREE of HEARTS

  SEVEN of HEARTS
+ SEVEN of CLUBS

You have the queen...Start!
You remove a card from the next player, Fruit (2). Give a position (1-9).
1
You remove a card from the next player,  Fruit (2) : JACK of CLUBS
  JACK of HEARTS
+ JACK of CLUBS

Your hand:
Card number: 01 : SEVEN of SPADES
Card number: 02 : FIVE of CLUBS
Card number: 03 : FOUR of SPADES
Card number: 04 : NINE of CLUBS
Card number: 05 : JACK of DIAMONDS
Card number: 06 : EIGHT of DIAMONDS
Card number: 07 : FIVE of DIAMONDS


Fruit (2) Take your card: FIVE of CLUBS

  FIVE of HEARTS
+ FIVE of CLUBS

Number of players is: 2
You have 6 cards
```

```
Fruit (2) a : 7 cards
Game starts on turn 2
You remove a card from the next player, Fruit (2). Give a position (1-7).
1
You remove a card from the next player,  Fruit (2) : NINE of HEARTS
  NINE of CLUBS
+ NINE of HEARTS

Your hand:
Card number: 01 : SEVEN of SPADES
Card number: 02 : FOUR of SPADES
Card number: 03 : JACK of DIAMONDS
Card number: 04 : EIGHT of DIAMONDS
Card number: 05 : FIVE of DIAMONDS


Fruit (2) Take your card: SEVEN of SPADES

  SEVEN of DIAMONDS
+ SEVEN of SPADES

Number of players is: 2
You have 4 cards
Fruit (2) a : 5 cards
Game starts on turn 3
You remove a card from the next player, Fruit (2). Give a position (1-5).
1
You remove a card from the next player,  Fruit (2) : FOUR of DIAMONDS
  FOUR of SPADES
+ FOUR of DIAMONDS

Your hand:
Card number: 01 : JACK of DIAMONDS
Card number: 02 : EIGHT of DIAMONDS
Card number: 03 : FIVE of DIAMONDS


Fruit (2) Take your card: EIGHT of DIAMONDS

  EIGHT of SPADES
+ EIGHT of DIAMONDS

Number of players is: 2
You have 2 cards
Fruit (2) a : 3 cards
Game starts on turn 4
You remove a card from the next player, Fruit (2). Give a position (1-3).
1
You remove a card from the next player,  Fruit (2) : JACK of SPADES
  JACK of DIAMONDS
+ JACK of SPADES

Your hand:
Card number: 01 : FIVE of DIAMONDS
```

```
Fruit (2) Take your card: FIVE of DIAMONDS

  FIVE of SPADES
+ FIVE of DIAMONDS

Number of players is: 2
You have 0 cards
Fruit (2) a : 1 cards
Game starts on turn 5
You are good!
You no longer have a card! Nice job!
Game finished on turn 6
You won after 6 turns!
Fruit (2) lost after 6 turns!
```