Lesson9

Work9-1（P244）
トランプゲームフレームワークを利用して、七並べプログラムを作成するのに必要なクラスとメソッドは何か？

追加するクラス
・ルールクラス
・テーブルクラス
・プレイヤークラス
・進行役クラス

追加するメソッド
・テーブルに置けるカードを探す（ルールクラス）
・パスを宣言する（進行役）
・カードを置く（テーブル）
・カードを見る（テーブル）
・パス回数を数える（プレイヤー）

※Work9-2から9-6まで、ソースコードが見にくい感じがしたのでテキストファイルもフォルダに入れて置きます…

Work9-2（P250）
FantanTable.javaを完成させる

```
package fantan;

import lesson8.Card;
import lesson8.Table;

/**
 * 七並べのテーブルクラス
 * @author yuna
 * @date 2023/6/21
 */
public class FantanTable implements Table{

	//テーブルに並んだカードを表す配列(4*13)
	private Card[][] tableCard = new Card[Card.SUIT_NUM][Card.CARD_NUM];

	/**
	 * カードを置く
	 * @param カード,持ってる手札を端から順に0,1,2,,,としてる
	 * @author yuna
	 * @date 2023/6/21
	 */
	@Override
	public void putCard(Card[] card) {
		//手札の1枚目のカードのスートを見る
		int suitNumber = card[0].getSuit();
		//手札の1枚目のカードの数字を見る
		int cardNumber = card[0].getCardNumber();
		/*
```

```
                * 手札をテーブルの適切な位置におく
                * 配列のインデックスは0からなので-1しておく*/
                tableCard[suitNumber-1][cardNumber-1] = card[0];
        }

        /**
         * カードを見る。
         * @return テーブルに置かれたカードを表す配列
         */
        @Override
        public Card[][] getCards() {
                //見せるようにテーブルのカード配列のコピーを作る
                Card[][] tableCopy = new Card[Card.SUIT_NUM][Card.CARD_NUM];
                //スートごとにループする（行ループ）
                for(int suit=0; suit<Card.SUIT_NUM; suit++) {
                        /*
                         * arraycopy(コピー元配列, コピー元配列のコピー開始位置,
                         * コピー先配列, コピー先配列の開始位置, コピーの個数)
                        */
                        System.arraycopy(tableCard[suit], 0, tableCopy[suit], 0,
Card.CARD_NUM);
                }
                //テーブルに置かれたカードを表す配列を返す
                return tableCopy;
        }

        /**
         * テーブルのカードを表す文字列表現
         * @return        テーブル上のカードの文字列表現を返却する
         */
        @Override
        public String toString() {
                //StringBunfferを宣言
                StringBuffer tableString = new StringBuffer();
                //スートのループ（行ループ）
                for(int suit=0 ; suit<Card.SUIT_NUM; suit++) {
                        //数字のループ（列ループ）
                        for(int num=0; num<Card.CARD_NUM; num++) {
                                //指定位置のカードがまだ置かれていないとき
                                if(tableCard[suit][num] == null) {
                                        //代わりに..を表示する
                                        tableString.append("..");
                                }
                                //置かれてた場合は
                                else {
                                        //その位置のカードのスートと数字を表示する
                                        tableString.append(tableCard[suit][num]);
                                }
                                //隣を表示するまえにスペースを追加する
                                tableString.append(" ");
                        }
                        //次の行に移るために改行をする
                        tableString.append("\n");
                }
                //文字列表現を返却する
                return tableString.toString();
        }

}
```

Work9-3（P255）

FantanRuleクラスの作成

```
package fantan;

import lesson8.Card;
import lesson8.Hand;
import lesson8.Rule;
import lesson8.Table;

/**
 * 七並べ用のルールクラス
 * @author yuna
 * @date 2023/6/22
 */
public class FantanRule implements Rule{

	/**
	 * テーブルに出せるカードを探す
	 * @param hand　手札
	 * @param Table テーブルに出てるカード
	 * @return Card[]　テーブルに出せるカードの組み合わせ、なければnullを返す
	 */
	@Override
	public Card[] findCandidate(Hand hand, Table table) {
		//手札の枚数を数える
		int numberOfCards = hand.getNumberOfCards();
		/*
		 * おけるカード候補を格納する配列
		 * おけるカードが無い場合はnullを返す*/
		Card[] findCards = null;

		//手札のカードを順に確認する
		for(int index=0; index<numberOfCards; index++) {
			//手札のカードを見る
			Card lookingCard = hand.lookCard(index);
			//見てるカードのスートを確認する
			int lookingSuit = lookingCard.getSuit();
			//見てるカードの数字を確認する
			int lookingNum  = lookingCard.getCardNumber();

			//見てるカードの右隣の数字
			int rightNum = lookingNum+1;
			//見てるカードの左隣の数字
			int leftNum = lookingNum-1;

			//見てるカードが１の場合
			if(lookingNum == 1) {
				//左隣のカードは13になる
				leftNum = 13;
			}
			//見てるカードが13の場合
			else if(lookingNum == 13) {
				//右隣のカードは1になる
				rightNum=1;
			}
			//置きたいカードと隣接するカードがテーブルにおかれている場合
			if(isThereCard(table,lookingSuit,rightNum)
				||isThereCard(table,lookingSuit,leftNum)) {
```

```
                                        //見てるカードをおけるカードの配列に格納する
                                        findCards = new Card[1];
                                        findCards[0] = hand.pickCard(index);
                                        break;
                                }
                        }
                        //出せるカードのうち最初に見つかったものを返却する
                        return findCards;
                }
                /**
                 * 引数で指定した位置のカードが置かれるかどうか調べる
                 * @param table テーブル
                 * @param cardSuit　スート
                 * @param cardNumber　数字
                 * @return　カードが置かれている場合はtrueを返す
                 */
                private boolean isThereCard(Table table, int cardSuit, int cardNumber) {
                        //テーブルにカードがなかった場合はfalseを返す
                        boolean result= false;
                        //テーブル上のカードを取得する
                        Card[][] tableCards = table.getCards();

                        //指定したスートと数の位置に配列要素がある場合
                        if(tableCards[cardSuit-1][cardNumber-1] != null) {
                                //カードがあった場合はtrue
                                result = true;
                        }
                        //カードがあったかどうかを返却する
                        return result;
                }
}
```

Work9-4（P261）

FantanPlayerクラスを完成させる

```
package fantan;
import lesson8.Card;
import lesson8.Player;

/**
 * 七並べのプレイヤークラス
 * @author yuna
 * @date 2023/6/22
 *
 */
public class FantanPlayer extends Player{

        //コンストラクタ
        FantanPlayer(String playerName, FantanMaster masterPlayer,
                        FantanTable table_, FantanRule trumpRule) {
                super(playerName,masterPlayer,table_,trumpRule);
        }

        //パスした回数を数える
        private int passCount = 0;

        /**
         * パスした回数を数えて返却する
```

```
	 * @return パスした回数を返却する
	 */
	public int getPassCount() {
		//パスした回数をカウントする、
		passCount++;
		//回数を返却する
		return passCount;
	}

	@Override	//これがplayメソッドです
	public void nextPlayer(Player nextPlayer) {
		System.out.println(" "+ myHand);
		//テーブルをみておけるカードを探す
		Card[] candidate = trumpRule.findCandidate(myHand,table_);
		//返値がnullでない場合(おけるカード有)
		if( candidate != null ) {
			System.out.println(playerName+":"+candidate[0] + "を置きました。\n");
			//テーブルにカードを置く
			table_.putCard(candidate );
			//手札が0枚になったら
			if(myHand.getNumberOfCards() == 0) {
				//上がり宣言をする
				masterPlayer.declareWin(nextPlayer);
			}
		}
		//そうでない場合
		else {
			//進行役にパスを宣言する
			((FantanMaster)masterPlayer).pass((FantanPlayer) nextPlayer);
			if(passCount>FantanMaster.PASSABLE_LIMIT) {
				//残りの手札の枚数を数える
				int NumberOfLeaveCard = myHand.getNumberOfCards();
				//持っていたカードをすべてテーブルに置く
				for(int index=0 ; index<NumberOfLeaveCard ; index++) {
					//手札からカードを一枚引く
					Card[] leaveCards = {myHand.pickCard(0)};
					//引いたカードを置く
					table_.putCard(leaveCards);
				}
			}
		}
		//テーブルの状態を表示する
		System.out.println(table_);
	}
	/**
	 * 進行役から配られたカードを手札に加える
	 * そのカードが 7 だった場合はテーブルに並べる
	 * @param card 手札として受け取ったカード
	 */
	@Override
	public void receiveCard(Card card) {
		//配られたカードをみる
		int cardNumber = card.getCardNumber();
		//数字が 7 の場合
		if(cardNumber == 7) {
			System.out.println(playerName + ":" + card + "を置きました。");
			//テーブルにカードを置く
			table_.putCard(new Card[] {card}) ;
		}
```

```
                    //そうでない場合は
                    else {
                            //カードを自分の手札に加える
                            super.receiveCard(card);
                    }
            }
}
```

Work9-5（P266）

mainメソッドに必要な処理を加えて七並べプログラムを完成させる。

```
package fantan;

import lesson8.Card;
import lesson8.Hand;
/**
 * @author yuna
 *
 */
public class test {

            /**
             * 七並べのテストクラス
             * @param ayuna
             */
            public static void main(String[]args) {
                    //進行役の生成
                    FantanMaster masterPlayer = new FantanMaster();
                    //テーブルの生成
                    FantanTable fantanTable   = new FantanTable();
                    //ルールの生成
                    FantanRule fantanRule = new FantanRule();
                    //プレイヤー生成
               FantanPlayer murata = new FantanPlayer("村田",masterPlayer,fantanTable,fantanRule);
               FantanPlayer yamada = new FantanPlayer("山田",masterPlayer,fantanTable,fantanRule);
               FantanPlayer saito  = new FantanPlayer("斎藤",masterPlayer,fantanTable,fantanRule);
                    //プレイヤーの登録
               masterPlayer.registerPlayer(murata);
               masterPlayer.registerPlayer(yamada);
               masterPlayer.registerPlayer(saito);
                    //トランプを生成する
                    Hand handTrump = createtrump();
                    //ゲームの準備をする
                    masterPlayer.prepareGame(handTrump);
                    //ゲームを開始する
                    masterPlayer.startGame();
            }

private static Hand createtrump() {
            //トランプを生成する
            Hand trump = new Hand();
            //数字の1から13まで
            for(int number= 1; number<=13 ; number++) {
                    //スートがクローバーのカードを生成してトランプの手札に加える
                    trump.addCard(new Card(Card.SUIT_CLOVER, number));
                    //スートがハートのカード
                    trump.addCard(new Card(Card.SUIT_HEART, number));
```

```
                //スートがダイヤのカード
                trump.addCard(new Card(Card.SUIT_DIAMOND, number));
                //スートがスペードのカード
                trump.addCard(new Card(Card.SUIT_SPADE, number));
        }
        //トランプを返却する（今回はジョーカーを含まない）
        return trump;
}
}
```

Work9-6（P270）

フレームワークを用いてババ抜きプログラムをもう一度作成する

OldMaidMasterクラス（Masterクラスを継承）

```
package OldMaid;
import lesson8.Card;
import lesson8.Hand;
import lesson8.Master;
import lesson8.Player;
/**
 * ババ抜き用の進行役クラス
 * @author yuna
 * @date 2023/6/22
 */
public class OldMaidMaster extends Master{
        /**
         * ゲームの準備をする
         * プレイヤーに順番にカードを配り、
         * 手札に同じで数字があればテーブルに捨てる
         * @param cards 進行役の手札
         */
        @Override
        public void prepareGame(Hand cards) {
                //ゲームの準備を始めることを表示する
                System.out.println("【カードを配ります】");
                //トランプをシャッフルする
                cards.shuffleCard();
                //トランプの枚数を取得する
                int numberOfCards = cards.getNumberOfCards();
                //プレイヤーの人数を取得する
                int numberOfPlayers = playersList.size();
                //手札がなくなるまでカードを配る
                for(int index = 0; index < numberOfCards; index++) {
                        //カードを一枚引く
                        Card card = cards.pickCard(0);
                        //カードを配る
                        Player player = (Player) playersList.get(index%numberOfPlayers);
                        //プレイヤーはカードを受け取り手札に同じ数があればテーブルに捨てる
                        player.receiveCard(card);
                }
        }
}
```

OldMaidPlayerクラス（Playerクラスを継承）

```
package OldMaid;

import lesson8.Card;
import lesson8.Hand;
import lesson8.Master;
import lesson8.Player;
/**
 * @author staff
 *
 */
public class OldMaidPlayer extends Player{
	//コンストラクタ
	OldMaidPlayer(String playerName, Master masterPlayer,
			OldMaidTable table_, OldMaidRule trumpRule)
	{
		super(playerName,masterPlayer,table_,trumpRule);
	}

	/**
	 * ババ抜きでプレイヤーが自分のターンに行うこと
	 * @param nextPlayer　次の順番のプレイヤー
	 */
	@Override
	public void play(Player nextPlayer) {
		//次のプレイヤーに手札を出してもらう
		Hand nextHand = ((OldMaidPlayer) nextPlayer).showHand();
		//相手の手札からカードを一枚引く
		Card pickedCard = nextHand.pickCard(0);
		//引いた結果を表示する
		System.out.println(this +"："+nextPlayer+"さんから"+pickedCard+"を引きました");
		//引いたカードを自分の手札に加える、同じ数字のカードは捨てる
		receiveCard(pickedCard);
		//手札の枚数を調べる
		//手札の枚数が 0 の場合
		if(myHand.getNumberOfCards() == 0) {
			//進行役に上がり宣言する
			masterPlayer.declareWin(this);
		}
		//そうでなければ現在の手札を確認する
		else {
			System.out.println(this+"：残りの手札は\n"+myHand+"です");
		}
	}
	/**
	 * カードを受け取る・または配る<br>
	 * 必要に応じてオーバーライドすること
	 * @param card 受け取ったカード
	 * @author yuna
	 */
	@Override
	public void receiveCard(Card card) {
		//カードを自分の手札に加える
		myHand.addCard(card);
		//今加えたカードと同じ数字のカードが無いか調べる
		Card[] sameCards = trumpRule.findCandidate(myHand,table_);
		//同じ数字のカードあった場合
		if(sameCards != null) {
			//誰がどのカードを捨てたかを表示する
			System.out.print(this+"：");
```

```
                        //カードをテーブルに捨てる
                        table_.putCard(sameCards);
                }
        }
        /**
         * @return  自分の手札を見せる
         */
        public Hand showHand() {
                //もし手札が残り一枚の場合
                if(myHand.getNumberOfCards() == 1) {
                        //進行役に上がり宣言する
                        masterPlayer.declareWin(this);
                }
                //見せる前にシャッフルする
                myHand.shuffleCard();
                //手札を見せる
                return myHand;
        }
}
```

OlaMaidRuleクラス

```
package OldMaid;
import lesson8.Card;
import lesson8.Hand;
import lesson8.Rule;
import lesson8.Table;

public class OldMaidRule implements Rule{
        /**
         * テーブルに捨てられるカードを探す<br>
         * 同じ数字のカードがあればその数字の格納された配列を返却し
         * @param hand                  手札
         * @param table                 テーブル
         * @return    見つかったカードの組み合わせ、見つからない時はnullを返す
         */
        @Override
        public Card[] findCandidate(Hand hand, Table table) {

                //手札の枚数を表す変数
                int numberOfCards = hand.getNumberOfCards() ;
                //数字の同じカードのリスト、同じカードがなかった時がnullとする
                Card [] findCard = null;

                //手札にカードが一枚も無い場合は何もしない
                if(numberOfCards > 0) {
                        //最後に追加されたカードのインデックスを取得する
                        int lastIndex = numberOfCards -1;
                        //最後に追加されたカードを取得する
                        Card lastAddCard = hand.lookCard(lastIndex);
                        //最後に追加されたカードの数字を取得する
                        int lastAddCardNumber = lastAddCard.getCardNumber();

                        //配列の先頭から末尾まで順番に走査する
                        for(int index=0; index<lastIndex; index++) {
                                //今見てるインデックスの位置のカードのインスタンスを生成する
                                Card card = hand.lookCard(index);
```

```
                                        //そのカードの数字が加えたカードの数字と一致した場合
                                        if(card.getCardNumber() == lastAddCardNumber) {
                                                //同じカードの配列を要素数 2 で宣言
                                                findCard = new Card[2];
                                                //配列に最後に加えたカードを手札から取り出して格納
                                                findCard[0] = (Card)hand.pickCard(lastIndex);
                                                //配列に見つけた数字の同じカードを手札から取り出す
                                                findCard[1] = (Card)hand.pickCard(index);
                                                //ループから抜ける
                                                break;
                                        }
                                }
                        }
                        return findCard;
                }
}
```

OladMaidTableクラス（Tableクラスを継承）

```
package OldMaid;
import java.util.ArrayList;
import lesson8.Card;
import lesson8.Table;
/**
 * ババ抜きのテーブルクラス
 * @author yuna
 * @date 2023/6/22
 */
public class OldMaidTable implements Table{
        //捨てられたカードを保持しておくためのリスト
        private ArrayList<Card[]> disposedCards = new ArrayList<Card[]>();
        /**
         * 捨てるカードを表示して配列に保持する
         * @param card 捨てるカード
         * @author yuna
         */
        @Override
        public void putCard(Card[] card) {
                //引数の配列の先頭から末尾までループする
                for (int index = 0; index < card.length; index++) {
                        //カードObjectのtoStringメソッドが自動で呼び出される
                        System.out.print(card[index]+" ");
                }
                //カードを捨てたことを表示する
                System.out.println("を捨てました");
                //捨てたカードを「捨てられたカードリスト」に追加する
                disposedCards.add(card);
        }
        /**
         * 捨てられたカードの並べ方
         * ババ抜きでカードは並べないので、nullを返すことにする
         */
        @Override
        public Card[][] getCards() {
                return null;
        }
}
```

OldMAidTestクラス（ババ抜きを実行するためのmainメソッド）

```
package OldMaid;
import lesson8.Card;
import lesson8.Hand;
import lesson8.Joker;
import lesson8.Master;

public class OldMaidTest {
	/**
	 * @param args
	 */
	public static void main(String[]args) {
		//進行役を生成
		Master masterPlayer  = new Master();
		//テーブルを生成
		OldMaidTable  field  = new OldMaidTable();
		//ババ抜きのルールを生成
		OldMaidRule oldMaidRule = new OldMaidRule();

		//プレイヤーの生成
	   OldMaidPlayer murata = new OldMaidPlayer("村田",masterPlayer,field,oldMaidRule);
	   OldMaidPlayer yamada = new OldMaidPlayer("山田",masterPlayer,field,oldMaidRule);
	   OldMaidPlayer saito  = new OldMaidPlayer("斎藤",masterPlayer,field,oldMaidRule);

	   //プレイヤーの登録
		masterPlayer.registerPlayer(murata);
		masterPlayer.registerPlayer(yamada);
		masterPlayer.registerPlayer(saito);

		//トランプを作る
		Hand trump = createTrump();
		//ゲームの準備をする
		masterPlayer.prepareGame(trump);
		//ゲームを開始する
		masterPlayer.startGame();
	}

	/**
	 * トランプを作る(カード52枚とジョーカー１枚)
	 * @return トランプのカードを返却する
	 */
	private static Hand createTrump() {
		//手札を生成する
		Hand trump = new Hand();
		//数字が１から１３まで順にカードを生成して手札に加えていく
		for(int number= 1; number<=13 ; number++) {
			//スートがクラブのカード
			trump.addCard(new Card(Card.SUIT_CLOVER, number));
			//スートがハートのカード
			trump.addCard(new Card(Card.SUIT_HEART, number));
			//スートがダイヤのカード
			trump.addCard(new Card(Card.SUIT_DIAMOND, number));
			//スートがスペードのカード
			trump.addCard(new Card(Card.SUIT_SPADE, number));
		}
		//ジョーカーを生成して手札に加える
		trump.addCard(new Joker());
		//トランプを返す
		return trump;
	}
```

```
}
```

実行画面

```
【カードを配ります】
村田：SJ CJ を捨てました
斎藤：SK HK を捨てました
山田：H6 C6 を捨てました
山田：S9 H9 を捨てました
村田：D5 C5 を捨てました
山田：H2 D2 を捨てました
山田：HA SA を捨てました
斎藤：S6 D6 を捨てました
村田：D4 H4 を捨てました
斎藤：H5 S5 を捨てました
山田：H3 S3 を捨てました
村田：D8 C8 を捨てました
村田：ST DT を捨てました
山田：D7 C7 を捨てました
斎藤：HQ SQ を捨てました
【ゲームを開始します】

村田さんの番です
村田：山田さんからCKを引きました
村田：残りの手札は
CA DQ JK D3 H7 S4 D9 HT CK です

山田さんの番です
山田：斎藤さんからDAを引きました
山田：残りの手札は
S2 CQ C3 DJ H8 DA です

斎藤さんの番です
斎藤：村田さんからCAを引きました
斎藤：残りの手札は
C2 C4 CT HJ S8 C9 DK S7 CA です

村田さんの番です
村田：山田さんからDJを引きました
村田：残りの手札は
CK D3 JK H7 D9 S4 DQ HT DJ です

山田さんの番です
山田：斎藤さんからDKを引きました
山田：残りの手札は
C3 H8 DA CQ S2 DK です

斎藤さんの番です
斎藤：村田さんからCKを引きました
斎藤：残りの手札は
CT S8 CA C2 HJ C4 C9 S7 CK です

村田さんの番です
村田：山田さんからDKを引きました
村田：残りの手札は
DJ S4 D3 D9 JK HT H7 DQ DK です
```

山田さんの番です
山田：斎藤さんからCKを引きました
山田：残りの手札は
C3 DA H8 S2 CQ CK です

斎藤さんの番です
斎藤：村田さんからDQを引きました
斎藤：残りの手札は
C9 CT S8 C2 HJ S7 C4 CA DQ です

村田さんの番です
村田：山田さんからH8を引きました
村田：残りの手札は
JK HT H7 D3 DK D9 S4 DJ H8 です

山田さんの番です
山田：斎藤さんからC9を引きました
山田：残りの手札は
DA CK C3 CQ S2 C9 です

斎藤さんの番です
斎藤：村田さんからS4を引きました
斎藤：S4 C4 を捨てました
斎藤：残りの手札は
S7 CT DQ C2 HJ S8 CA です

村田さんの番です
村田：山田さんからCQを引きました
村田：残りの手札は
D3 H7 D9 HT DK DJ JK H8 CQ です

山田さんの番です
山田：斎藤さんからCTを引きました
山田：残りの手札は
DA S2 C9 CK C3 CT です

斎藤さんの番です
斎藤：村田さんからD9を引きました
斎藤：残りの手札は
CA S8 S7 C2 DQ HJ D9 です

村田さんの番です
村田：山田さんからS2を引きました
村田：残りの手札は
H8 HT JK D3 DK H7 CQ DJ S2 です

山田さんの番です
山田：斎藤さんからDQを引きました
山田：残りの手札は
C3 C9 CT DA CK DQ です

斎藤さんの番です
斎藤：村田さんからHTを引きました
斎藤：残りの手札は
CA S8 C2 D9 S7 HJ HT です

村田さんの番です
村田：山田さんからCKを引きました
村田：CK DK を捨てました

村田：残りの手札は
D3 JK S2 H8 CQ DJ H7 です

山田さんの番です
山田：斎藤さんからS7を引きました
山田：残りの手札は
C3 DA CT C9 DQ S7 です

斎藤さんの番です
斎藤：村田さんからCQを引きました
斎藤：残りの手札は
HJ HT CA S8 D9 C2 CQ です

村田さんの番です
村田：山田さんからC3を引きました
村田：C3 D3 を捨てました
村田：残りの手札は
H7 H8 JK DJ S2 です

山田さんの番です
山田：斎藤さんからD9を引きました
山田：D9 C9 を捨てました
山田：残りの手札は
S7 CT DQ DA です

斎藤さんの番です
斎藤：村田さんからH8を引きました
斎藤：H8 S8 を捨てました
斎藤：残りの手札は
CA CQ C2 HJ HT です

村田さんの番です
村田：山田さんからDQを引きました
村田：残りの手札は
DJ S2 JK H7 DQ です

山田さんの番です
山田：斎藤さんからCAを引きました
山田：CA DA を捨てました
山田：残りの手札は
S7 CT です

斎藤さんの番です
斎藤：村田さんからJKを引きました
斎藤：残りの手札は
C2 CQ HJ HT JK です

村田さんの番です
村田：山田さんからS7を引きました
村田：S7 H7 を捨てました
村田：残りの手札は
S2 DJ DQ です

山田さんの番です
山田：斎藤さんからHTを引きました
山田：HT CT を捨てました
山田さんが上がりました！

斎藤さんの番です

斎藤：村田さんからDJを引きました
斎藤：DJ HJ を捨てました
斎藤：残りの手札は
CQ C2 JK です

村田さんの番です
村田：斎藤さんからJKを引きました
村田：残りの手札は
DQ S2 JK です

斎藤さんの番です
斎藤：村田さんからDQを引きました
斎藤：DQ CQ を捨てました
斎藤：残りの手札は
C2 です

村田さんの番です
斎藤さんが上がりました！
村田さんの負けです
村田：斎藤さんからC2を引きました
村田：C2 S2 を捨てました
村田：残りの手札は
JK です
【ゲームを終了しました】