卒業研究報告書

題目

セブンブリッジにおける高い勝率を得るためのプレイングとは

指導教員

石水　隆　講師

報告者

11-1-037-0071

松村 幸俊

近畿大学理工学部情報学科

平成 28 年 1 月 29 日提出

概要

セブンブリッジは麻雀と同じラミー系というゲーム分野に属す不完全情報ゲームである．不完全情報ゲームは不確実な情報のもとて相乎の手の強さやケーム戦略を推論しなけれはならす、その推譲結果には不確実性か伴い完全な解析が不可能な為必勝法が存在しない．[1]

本研究では、セブンブリッジにおける高い勝率を得るための戦略について検証する。本研究ではセブンブリッジにおいて有利にプレイする為の基本的な戦略を追求するために，異なる戦略に基づいてセブンブリッジをプレイするAIを複数実装しそれらを対戦させて検証する．

目次

1. 序論

1.1. 本研究の背景

近年，将棋やオセロ等のボードゲームの思考アルゴリズムの開発や人工知能（AI）の開発，またそれらと人間のプレイヤーとの対戦等の話題を頻繁に耳にする．特に将棋の電王戦[2]は記憶に新しい．これらは“二人零和有限確定完全情報ゲーム”と呼ばれる．これに分類されるゲームは理論上はすべての手を先読みすることが可能であり，双方のプレイヤーが最善手を打てば必ず先手必勝，後手必勝，引き分けのいずれかに決定される．

それに対しセブンブリッジや麻雀等は“零和有限不確定非完全情報ゲーム”に分類される．これらは“ランダムに積まれた山の中から一枚引く”等の偶然の要素が入り込み，さらに他人の手札等，状況が完全に把握できない要素が含まれる．完全情報ゲームではゲーム終了時の情報から逆算し，有利な状況を導き出すことが可能だが，不完全情報ゲームでは他のプレイヤーの状態が分からないため先読みを行うことが非常に困難になる．このため、一般に不完全情報ゲームでは必勝法は祖納せず、従って思考アルゴリズムの開発は困難であり、ゲームAIの強さにも限界がある．

1.2. 既知の結果

完全情報ゲームは探索木を用いて最善手を決定しており，理論上は全ての手を探索することが可能であるため必勝手が存在する．必勝手でプレイすれば負けることはなく，先手か後手が決まった時点で勝敗が決まるか引き分けになる．不完全情報ゲームではプレイヤーが得られる情報が限られているため，そこから推測される情報も不完全なものとなる．そのため，ゲームを有利に進めるための最善手は存在するが，完全情報ゲームのような必勝手は存在しない．[3]

1.3. 本研究の目的

前節で述べたように，不確定非完全情報ゲームには必勝方法は存在しない．したがって，不完全ゲームにおいては、より勝率の高い手を選択することがゲームAIの目標となる．本研究ではセブンブリッジにおいて，高い勝率を得ることが出来る基本的な打ち方を探す．そのためにおおまかな戦略アルゴリズムを実装したAIを使い，どのような傾向でプレイするのが有効なのかを導き出す．

1.4. 本報告書の構成

２章１節以降の各節の内容を簡潔に記述する．２章ではセブンブリッジについての概要．３章では作成したプログラムとAIについての説明．４章ではそれらを用いて導き出された結果と，それについての考察を行う．５章では以上の事を全て踏まえた上で，今後の課題について考えた事を述べる．

2. セブンブリッジとは

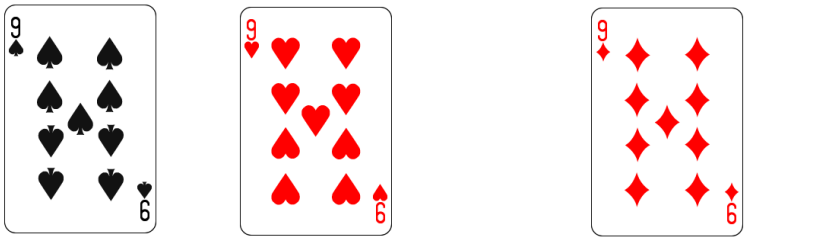
本項ではセブンブリッジのゲームについて簡単に説明する．プレイする環境によって細かな違いがあるが，本研究ではこれから記述するルールに則って研究を行う．また以降では，カードは“H4”のようにスートと数字の組み合わせで表記する．ただし、C,D,H,S はそれぞれクラブ，ダイヤモンド，ハート，スペードを表す。

2.1. セブンブリッジのルール

セブンブリッジは２～６人で行うトランプを使用するゲームであり，ジョーカーを除く５２枚のカードを用いてプレイする．ジョーカーを用いる場合は５３枚で行いオールマイティとして扱う．ただし本研究ではジョーカーは使用しない．

ラウンド開始時，それぞれのプレイヤーに手札が７枚配られる．毎ターン山札を１枚引きメルドを行う．フィールドに手札を出す行為をメルドという．詳細は2.2節を参照．メルドが不可能もしくはしなくて良いと判断した場合，手札を１枚捨て次のプレイヤーのターンにうつる．

麻雀と同じ鳴きのルールがあり，本ゲームにはポンとチーが存在する。ポンは他のプレイヤーが捨てた札と合わせた際に同位の札が３枚以上あれば行える。チーは自分の前の手番の人が捨てた札と合わせて３枚以上の同スートで連続した数の組が作れる場合に行える． 鳴きを行った場合は，捨てられたカードと共にフィールドに出す．図1，図2に鳴きを行える場合の例を示す．



手札にあるカード　　他のプレイヤーから捨てられたカード

図1 ポンが可能な場合の例

図1では，S9 と H9 が手札にあるので，他のプレイヤーから捨てられた D9 を利用してポンをすることが可能である．

手札にあるカード　　1つ前のプレイヤーから捨てられたカード

図2 チーが可能な場合の例

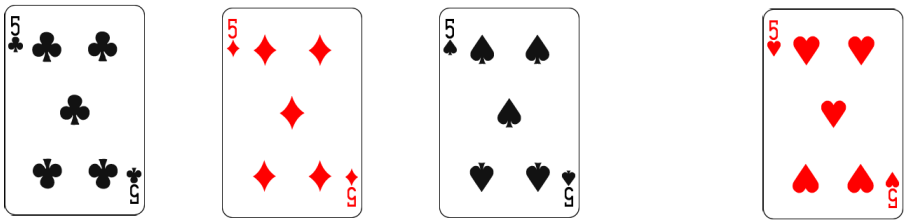
図2ではCJとCQが手札にあるので，1つ前のプレイヤーから捨てられたK9を利用してチーをすることが可能である．

誰か１人の手札が無くなるもしくは山札が無くなった時点で１ラウンドが終了し点数計算をする．

１０ラウンドを１ゲームとし，１ゲームが終了した時点での得点によって順位を決定する．セブンブリッジでは得点が低いプレイヤーが勝者となる．

2.2. メルドについて

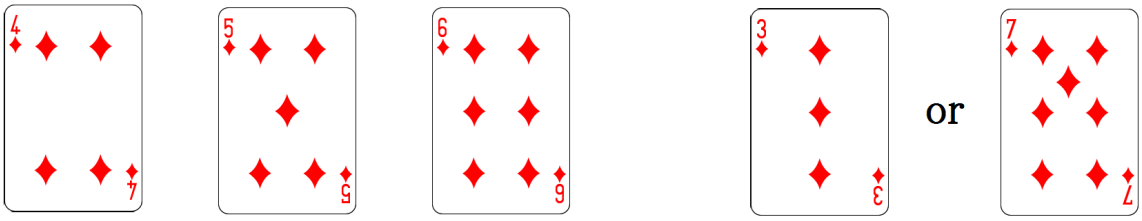
自分の手番ではメルドという行為があり，手札をフィールドに出すことが出来る．基本は同位の札を３枚以上，もしくは３枚以上の同スートで行うことができるが，７は１枚でもメルドすることが可能である．また，すでにフィールドに出されているカードに付加することも可能であり，その場合は数字に関わらず１枚で出すことが可能である．ただし同位のグループにスートで付加することはできない。逆も同じである．図3，図4にメルドが可能な場合の例を示す．



フィールドに出ているカード　　メルド可能なカード

図３ メルド可能な場合の例(同位札)

図3ではC5，D5，S5が場に出ているので，H5のみがメルド可能なカードである．



フィールドに出ているカード　　メルド可能なカード

図５ メルド可能な場合の例(同スート)

図4ではD4，D5，D6がフィールドに出ているので，メルド出来るカードはD3，D7の２種類になる．

2.3. 点計算について

ラウンド終了時に手札にあるカードを用いて点数の計算を行う．１～１０までの数札はその数字の点数，１１～１３までの絵札は１０点と計算する．セブンブリッジの特徴的なルールとして，ラウンド終了時に７を持っていると点数が２倍になるというルールが存在し，手札のカードを全て合計した後，その数値を７の枚数分２倍する．また今回は使用していないが，ジョーカーがあった場合点数が１０倍になる．表１に得点をまとめる

表1 各カードの点数

カード	点数
絵札	10点
数札(7以外)	それぞれの数字通り
7	1枚につき点数を2倍する
ジョーカー	点数を10倍する

2.4. 7のカードの重要性

ゲームの名前の通り，本ゲームにおける7のカードは特別な存在である．1枚でメルドすることが可能であり，自分の手を進めることが出来るというメリットはあるが，それとともに他のプレイヤーもメルドを行うことが容易になってしまうというデメリットもある．また，7をあまり出さない場合は他のプレイヤーがメルドし辛くなるため，手の進行を大きく阻害する事もメリットの一つだが，その反面，他のプレイヤーが上がってしまった場合点数が大幅に増えるというデメリットがある．このように非常に使い方が重要になってくるカードである．

3. 研究内容

3.1. セブンブリッジの戦略

本節では、セブンブリッジの戦略について述べる．

セブンブリッジでは他のプレイヤーが上がった時点での自分の手札に応じて点数が加算される．そのため大きな数字をラウンド終盤まで持ち続けないことがポイントとなる．不完全情報ゲームのため，それが必ずしも有効手になるとは限らない．だが点数の増加を可能な限り抑えるプレイングを行うことで，負けにくくなるというのが重要になっている．

本研究では、戦略として以下に示す3つの戦略を用いる．

戦略１：積極的にメルド・鳴きを行う戦略．早上がりの重要性を確認する．

戦略２：手札が４枚以下にできるまではメルドを行わない．相手の手の進行を阻害する行為の重要性を確認する．

戦略３：７は１枚では使用しない戦略．本ゲームでの７の重要性を確認する．

捨札の選択については，各戦略で同一のものを使用する．同位札やスートになっているものを除く，すなわち１枚で孤立しているカードを探し，その中で一番大きいカードを選択する．１枚で孤立しているものがなければ２枚でセットになっているカードの中から一番大きい数字を選択する．本ゲームはゲーム終了時のカードの数が点数になるため，手札の情報だけで見た場合大きい数字は積極的に減らして行くのが良いと考えられるため，このような戦略をとっている．

3.2. セブンブリッジプログラム

セブンブリッジの戦略を検証するに先立ち、本研究では Java を用いてセブンブリッジプログラムを作成した．付録にプログラムのソースを示す．

ゲームは全てコンソール上で行い，プログラム内で指定したラウンド数が行われると結果を表示して停止する．プログラムはそれぞれに鳴きやメルドのルールを与えており，条件が満たされた場合その行動を行う．

本研究で作成したセブンブリッジプログラムは、3.1節で述べた戦略ごとに１つのクラスを用いている．以下各クラスについて説明する．

3.2.1. P_Meld_Heavy クラス

戦略クラス１．積極的にメルドを行うクラス．メルドが可能なものは全て行い，鳴きが可能な場合は必ず行う．上がることで自分には点が入らないというゲームルールにおいて，早上がりがどれだけ有効であるかをこの戦略で判断する．

3.2.2. P_Meld_Light クラス

戦略クラス２．手札が４枚以下になるまではチーとスートでのメルドを行わない．手札が４枚以下になった後は鳴き，メルド共に積極的に行う．相手の手を進めない事がどれだけ重要であるかを判断する．同位札の場合手に持っていても相手の手の進行を妨害できるわけでは無いため，ポンや同位札を用いたメルドは最初から積極的に行う．判定はメルド可能なものをカウントし４枚以上メルドが可能なら実行する．

3.2.3. P_Seven_Double クラス

戦略クラス３．７は１枚ではメルドしない．２枚以上で行える場合のみメルドする．前述したように，本ゲームの７のカードは特別な存在である．７を容易にメルドしないことで他のプレイヤーの手を進めず，自分が有利に進めているのかどうかを判断する．

3.2.4. Game クラス

実際にゲームを実行するクラス．ゲームの流れとゲーム数，ラウンド数の設定をしている． while 文でゲームを実行し，その中で各クラスに与えたメソッドを呼び出してゲームを進行する．終了判定が出たら break を行い結果を表示し終了する．player_number という変数があり，現在のプレイヤーが誰であるかを保持している．

3.2.5. Playerクラス

プレイヤーの情報を保持するクラス．捨札の選択もこのクラスで行っている．手札を引く drawCard(int)メソッド，手札を表示する showHand(int)メソッド，手札を捨てる trashHand(int)メソッド．これらは引数の int 型変数でどのプレイヤーが行うかを指定している．

3.2.6. GMクラス

ゲームマスターの役割を行うクラス．プレイヤーに手札を配るメソッドやラウンドごとに各変数をクリアするためのメソッド，ゲームの終了判定等を行う．

3.2.7. Fieldクラス

フィールドの状態を保持するクラス．フィールドに出されているグループの型が同位札型なのか同スート型なのか，また何枚のグループであるかを保持する配列 int mgc[][]があり型を取得するための getMeldType(int)クラスがある．

3.2.8. Deckクラス

デッキを生成，シャッフルするクラス．デッキはアレイリストで作成し，Collections をインポートし shuffle を使用してシャッフルを行っている．

3.2.9. Pongクラス

ポンのルールを記述しているクラス．捨てられたカードと同位札のものを各プレイヤーから探し捨てられたカードを含め３枚以上存在した場合はポンが可能であると判定している．

3.2.10. Chowクラス

チーのルールを記述しているクラス．捨てられたカードと同スートになっているものを次のプレイヤーから探し，捨てられたカードを含め３枚以上存在した場合はチーが可能であると判定している．

3.2.11. MeldRuleクラス

カードを１枚のみでメルドする際のルールを記述しているクラス．複数枚で行うメルドはそれぞれの戦略クラスに記述している．７をメルドするタイミングや１枚でメルドを行うかどうかは各プレイヤーによって違う為，まずはプレイヤーごとに戦略に沿った判定をさせる．行う場合はフィールドのグループを１つづつ確認し，それが何型で何枚あるのかを取得する．その後同位札型の場合は数字を確認し同じ数字が手札にあればメルド，スート型の場合はそのグループの１枚目と最後のカードを確認しそれに続く形の数字があればメルドを行う．

4. 結果・考察

本研究では，前章で述べた３種類のAIを対戦させ１００ゲーム（１０００ラウンド）行った．

4.1. 対戦結果

対戦結果を表３に示す．

１００ゲームを行った際，順位の項目はそれぞれのプレイヤーが何度何位になったかを表している．合計得点は１００ゲームすべて行った後の最終得点である．

表３：対戦結果

		戦略１：Meld_Heavy	戦略２：Meld_Light	戦略３：Seven_Double
１位		96	3	1
２位		2	57	41
３位		2	40	58
合計点数		4694	24366	27008

4.2. 考察

以上の結果より本研究では戦略１の積極的に鳴き，メルドを行うのが非常に有効だということが分かる．２と３の戦略は安易なメルドを避け，相手の手の進行を遅らせる事が基本となっている．このことから手札を持ち続けるという行為は大きなリスクになると考えられる．自分の点数を増やさないようにプレイする事が非常に重要であることが分かる．

5. 結論・今後の課題

本研究では，セブンブリッジにおいて有利にプレイする為の基本的な戦略を追求した．手札が多いほどリスクが大きくなる本ゲームでは，いかに早く手札を減らすことができるかが１番に重要になっている．また本ゲームは不確定非完全情報ゲームであり，他プレイヤーの手札等はわからない．それを阻害するプレイングを行う事は非常に高度な思考を要する．

捨札や残りの山札や自分の手札等から見た場の状況を踏まえて相手の行動を阻害したり，手札が配られた時点で上がれる可能性が高いかどうかを判断し，可能な限り点数を減らす．そのような複雑な行動を行えるAIを作成することが今後の課題である．そうすることで更に勝率の高いプレイングを追求することが可能になる．

謝辞

本研究に際して，近畿大学理工学部情報学科情報論理工学研究室　石水隆講師にはお忙しい中研究のサポートをしていただき，論文指導に適切なご助言をいただきました．大変お世話になりました．

参考文献

[1] 鬼沢武久，風見覚，高橋千晴，不完全情報ゲームプレイングシステムの構築－スタッドポーカーを例にしてー，知能と情報 Vol.15，No.1，pp.127-141，日本知能情報ファジイ学会,.（2003），http://ci.nii.ac.jp/naid/110002690815/

[2] 将棋電王戦FINAL,http://ex.nicovideo.jp/denou/final/

[3] 津久井祐一，不完全情報ゲームにおける推論，研究報告ゲーム情報学(GI)，Vol.2004-GI-11，pp.1-2，情報処理学会，(2004). http://id.nii.ac.jp/1001/00058551/

[4] 西野順二，西野哲朗，コンピュータ大貧民における最良手の推定について，研究報告数理モデル化と問題解決(MPS)，Vol.2012-MPS-90，No,4，pp.1-6，情報処理学会，(2012). http://id.nii.ac.jp/1001/00083717/

[5] 根本佳典，古宮嘉那子，小谷善行，CRFを用いた麻雀の不完全情報推定，ゲームプログラミングワークショップ2012論文集，Vol.2012，No.6，pp.155-158，情報処理学会，(2012)，http://id.nii.ac.jp/1001/00091346/

付録

本研究で作成したセブンブリッジプログラムのソースを以下に示す.

Chowクラス

```
package sevsen_bridge;

import java.util.ArrayList;
import java.util.List;

public class Chow {

	static int can_chow = 0;//チーが可能かどうかを書くのする変数
	//捨てられたカードより小さいカード２枚を格納する為のリスト
	static List<String> c1_12 = new ArrayList<String>();
	//捨てられたカードより小さいカード１枚と大きいカード１枚を格納する為のリスト
	static List<String> c1_13 = new ArrayList<String>();
	//捨てられたカードより大きいカード２枚を格納する為のリスト
	static List<String> c1_23 = new ArrayList<String>();
	static int max; //コンピュータは一番大きい手を出すようにするため,これでリストの選択をする.

	public static void chow(int p_number , String trash_card) {
		List<String> chow_list = new ArrayList<String>();
		String card_mark = trash_card.substring(0,1);//文字列の0文字目から1文字目までを取り出す.
		String str2 = trash_card.substring(2,4);

		c1_12.add(trash_card);
		c1_13.add(trash_card);
		c1_23.add(trash_card);

		//P1
		if(p_number == 3){
			for (int a = 0 ; a < P_Meld_Heavy.hand.size() ; a++){
				if(P_Meld_Heavy.hand.get(a).startsWith(card_mark)){
					chow_list.add(P_Meld_Heavy.hand.get(a));
				}
			}

			chow_rule(str2 , chow_list , c1_12 , c1_13 , c1_23);


			if(can_chow == 1){

				int input_int = 1;

				if(input_int == 1){
					Game.c_judge = 1;
```

```
                                        if(max == 1){
                                                GM.meld_sort(cl_12);
                                                for(int chow = 0 ; chow < cl_12.size() ; chow++){

Field.meld_field [Field.meld_group_count][chow] = GM.tmp_meld_list.get(chow);
Field.mgc[Field.meld_group_count][0]++;
Field.mgc[Field.meld_group_count][1] = 2;

            while(P_Meld_Heavy.hand.remove(cl_12.get(chow))){};
                                                }
                                                Field.trash_card = "null";
                                                Field.meld_group_count++;
                                        }
                                        if(max == 2){
                                                GM.meld_sort(cl_13);

for(int chow = 0 ; chow < cl_13.size() ; chow++){

            Field.meld_field [Field.meld_group_count][chow] = GM.tmp_meld_list.get(chow);
            Field.mgc[Field.meld_group_count][0]++;
            Field.mgc[Field.meld_group_count][1] = 2;

            while(P_Meld_Heavy.hand.remove(cl_13.get(chow))){};
                                                }
                                                Field.trash_card = "null";
                                                Field.meld_group_count++;
                                        }
                                        if(max == 3){
                                                GM.meld_sort(cl_23);

for(int chow = 0 ; chow < cl_23.size() ; chow++){

            Field.meld_field [Field.meld_group_count][chow] = GM.tmp_meld_list.get(chow);
            Field.mgc[Field.meld_group_count][0]++;
            Field.mgc[Field.meld_group_count][1] = 2;

            while(P_Meld_Heavy.hand.remove(cl_23.get(chow))){};
                                                }
                                                Field.trash_card = "null";
                                                Field.meld_group_count++;
                                        }

                                        if(P_Meld_Heavy.hand.size() != 0){
                                                Player.trashHand(0);
                                        }
                                        Game.player_number = 0;
```

```
                    }
                }
            }

            //P2
            if(p_number == 0){
                for (int a = 0 ; a < P_Meld_Light.hand.size() ; a++){
                    if(P_Meld_Light.hand.get(a).startsWith(card_mark)){
                        chow_list.add(P_Meld_Light.hand.get(a));
                    }
                }

                chow_rule(str2 , chow_list , cl_12 , cl_13 , cl_23);


                if(can_chow == 1){
                    int input_int = 0;

                    if(P_Meld_Light.hand.size() <= 4 || P_Meld_Light.hand.size() -
chow_list.size() <= 4){
                        input_int = 1;
                    }

                    if(input_int == 1){
                        Game.c_judge = 1;

                        if(max == 1){
                            GM.meld_sort(cl_12);
            for(int chow = 0 ; chow < cl_12.size() ; chow++){
                Field.meld_field [Field.meld_group_count][chow] = GM.tmp_meld_list.get(chow);
                Field.mgc[Field.meld_group_count][0]++;
                Field.mgc[Field.meld_group_count][1] = 2;

        while(P_Meld_Light.hand.remove(cl_12.get(chow))){};
                            }
                            Field.trash_card = "null";
                            Field.meld_group_count++;
                        }
                        if(max == 2){
                            GM.meld_sort(cl_13);
            for(int chow = 0 ; chow < cl_13.size() ; chow++){
                Field.meld_field [Field.meld_group_count][chow] = GM.tmp_meld_list.get(chow);
                Field.mgc[Field.meld_group_count][0]++;
                Field.mgc[Field.meld_group_count][1] = 2;

        while(P_Meld_Light.hand.remove(cl_13.get(chow))){};
                            }
```

```
                                        Field.trash_card = "null";
                                        Field.meld_group_count++;
                                }
                                if(max == 3){
                                        GM.meld_sort(c1_23);
        for(int chow = 0 ; chow < c1_23.size() ; chow++){
                Field.meld_field [Field.meld_group_count][chow] = GM.tmp_meld_list.get(chow);
                Field.mgc[Field.meld_group_count][0]++;
                Field.mgc[Field.meld_group_count][1] = 2;

while(P_Meld_Light.hand.remove(c1_23.get(chow))){};
                                        }
                                        Field.trash_card = "null";
                                        Field.meld_group_count++;
                                }
                                if(P_Meld_Light.hand.size() != 0){
                                        Player.trashHand(1);
                                }
                                Game.player_number = 1;
                        }
                }
        }


        //P3
        if(p_number == 2){
                if(P_Seven_Double.hand.size() <= 4){
                        for (int a = 0 ; a < P_Seven_Double.hand.size() ; a++){
                                if(P_Seven_Double.hand.get(a).startsWith(card_mark)){
                                        chow_list.add(P_Seven_Double.hand.get(a));
                                }
                        }
                        chow_rule(str2 , chow_list , c1_12 , c1_13 , c1_23);


                        if(can_chow == 1){

                                int input_int = 1;
                                if(input_int == 1){
                                        Game.c_judge = 1;

                                        if(max == 1){
                                                GM.meld_sort(c1_12);
                        for(int chow = 0 ; chow < c1_12.size() ; chow++){
                        Field.meld_field [Field.meld_group_count][chow] = GM.tmp_meld_list.get(chow);
                        Field.mgc[Field.meld_group_count][0]++;
```

```
            Field.mgc[Field.meld_group_count][1] = 2;

            while(P_Seven_Double.hand.remove(cl_12.get(chow))){};
                                                                    }
                                                                    Field.trash_card = "null";
                                                                    Field.meld_group_count++;
                                                            }
                                                            if(max == 2){
                                                                    GM.meld_sort(cl_13);
                                                                    for(int chow = 0 ; chow <
cl_13.size() ; chow++){
                                                                            Field.meld_field
[Field.meld_group_count][chow] = GM.tmp_meld_list.get(chow);

            Field.mgc[Field.meld_group_count][0]++;

            Field.mgc[Field.meld_group_count][1] = 2;

            while(P_Seven_Double.hand.remove(cl_13.get(chow))){};
                                                                    }
                                                                    Field.trash_card = "null";
                                                                    Field.meld_group_count++;
                                                            }
                                                            if(max == 3){
                                                                    GM.meld_sort(cl_23);
                                                                    for(int chow = 0 ; chow <
cl_23.size() ; chow++){
                                                                            Field.meld_field
[Field.meld_group_count][chow] = GM.tmp_meld_list.get(chow);

            Field.mgc[Field.meld_group_count][0]++;

            Field.mgc[Field.meld_group_count][1] = 2;

            while(P_Seven_Double.hand.remove(cl_23.get(chow))){};
                                                                    }
                                                                    Field.trash_card = "null";
                                                                    Field.meld_group_count++;
                                                            }
                                                            if(P_Seven_Double.hand.size() != 0){
                                                                    Player.trashHand(3);
                                                            }
                                                            Game.player_number = 2;
                                                    }
                                            }
                                    }
                            }
```

```
                chow_list.clear();
                cl_12.clear();
                cl_13.clear();
                cl_23.clear();
                can_chow = 0;
                max = 0;
                //}
        }

        //チーのルールを記載しているメソッド　これを用いてcl_12,cl_13,cl_23にカードを入れていく.
        public static void chow_rule(String card_number , List<String> chow_list,
                        List<String> cl_12, List<String> cl_13, List<String> cl_23){

                if(card_number.equals("01")){
                        for (int a = 0 ; a < chow_list.size() ; a++){
                                if(chow_list.get(a).endsWith("02") || chow_list.get(a).endsWith("03") )
{
                                        cl_23.add(chow_list.get(a));
                                }
                        }
                }

                if(card_number.equals("02")){
                        for (int a = 0 ; a < chow_list.size() ; a++){
                                if(chow_list.get(a).endsWith("01") || chow_list.get(a).endsWith("03") )
{
                                        cl_13.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("03") || chow_list.get(a).endsWith("04") )
{
                                        cl_23.add(chow_list.get(a));
                                }
                        }
                }

                if(card_number.equals("03")){
                        for (int a = 0 ; a < chow_list.size() ; a++){
                                if(chow_list.get(a).endsWith("01") || chow_list.get(a).endsWith("02") )
{
                                        cl_12.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("02") || chow_list.get(a).endsWith("04") )
{
                                        cl_13.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("04") || chow_list.get(a).endsWith("05") )
{
```

```
                                        cl_23.add(chow_list.get(a));
                                }
                        }
                }

                if(card_number.equals("04")){
                        for (int a = 0 ; a < chow_list.size() ; a++){
                                if(chow_list.get(a).endsWith("02") || chow_list.get(a).endsWith("03") )
{
                                        cl_12.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("03") || chow_list.get(a).endsWith("05") )
{
                                        cl_13.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("05") || chow_list.get(a).endsWith("06") )
{
                                        cl_23.add(chow_list.get(a));
                                }
                        }
                }

                else if(card_number.equals("05")){
                        for (int a = 0 ; a < chow_list.size() ; a++){
                                if(chow_list.get(a).endsWith("03") || chow_list.get(a).endsWith("04") )
{
                                        cl_12.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("04") || chow_list.get(a).endsWith("06") )
{
                                        cl_13.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("06") || chow_list.get(a).endsWith("07") )
{
                                        cl_23.add(chow_list.get(a));
                                }
                        }
                }

                if(card_number.equals("06")){
                        for (int a = 0 ; a < chow_list.size() ; a++){
                                if(chow_list.get(a).endsWith("04") || chow_list.get(a).endsWith("05") )
{
                                        cl_12.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("05") || chow_list.get(a).endsWith("07") )
{
```

```
                                        cl_13.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("07") || chow_list.get(a).endsWith("08") )
{
                                        cl_23.add(chow_list.get(a));
                                }
                        }
                }

                if(card_number.equals("07")){
                        for (int a = 0 ; a < chow_list.size() ; a++){
                                if(chow_list.get(a).endsWith("05") || chow_list.get(a).endsWith("06") )
{
                                        cl_12.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("06") || chow_list.get(a).endsWith("08") )
{
                                        cl_13.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("08") || chow_list.get(a).endsWith("09") )
{
                                        cl_23.add(chow_list.get(a));
                                }
                        }
                }

                if(card_number.equals("08")){
                        for (int a = 0 ; a < chow_list.size() ; a++){
                                if(chow_list.get(a).endsWith("06") || chow_list.get(a).endsWith("07") )
{
                                        cl_12.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("07") || chow_list.get(a).endsWith("09") )
{
                                        cl_13.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("09") || chow_list.get(a).endsWith("10") )
{
                                        cl_23.add(chow_list.get(a));
                                }
                        }
                }

                if(card_number.equals("09")){
                        for (int a = 0 ; a < chow_list.size() ; a++){
                                if(chow_list.get(a).endsWith("07") || chow_list.get(a).endsWith("08") )
{
```

```
                                        cl_12.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("08") || chow_list.get(a).endsWith("10") )
{
                                        cl_13.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("10") || chow_list.get(a).endsWith("11") )
{
                                        cl_23.add(chow_list.get(a));
                                }
                        }
                }

                if(card_number.equals("10")){
                        for (int a = 0 ; a < chow_list.size() ; a++){
                                if(chow_list.get(a).endsWith("08") || chow_list.get(a).endsWith("09") )
{
                                        cl_12.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("09") || chow_list.get(a).endsWith("11") )
{
                                        cl_13.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("11") || chow_list.get(a).endsWith("12") )
{
                                        cl_23.add(chow_list.get(a));
                                }
                        }
                }

                if(card_number.equals("11")){
                        for (int a = 0 ; a < chow_list.size() ; a++){
                                if(chow_list.get(a).endsWith("09") || chow_list.get(a).endsWith("10") )
{
                                        cl_12.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("10") || chow_list.get(a).endsWith("12") )
{
                                        cl_13.add(chow_list.get(a));
                                }
                                if(chow_list.get(a).endsWith("12") || chow_list.get(a).endsWith("13") )
{
                                        cl_23.add(chow_list.get(a));
                                }
                        }
                }
```

```
                    if(card_number.equals("12")){
                              for (int a = 0 ; a < chow_list.size() ; a++){
                                        if(chow_list.get(a).endsWith("10") || chow_list.get(a).endsWith("11") )
{
                                                  cl_12.add(chow_list.get(a));
                                        }
                                        if(chow_list.get(a).endsWith("11") || chow_list.get(a).endsWith("13") )
{
                                                  cl_13.add(chow_list.get(a));
                                        }
                              }
                    }

                    if(card_number.equals("13")){
                              for (int a = 0 ; a < chow_list.size() ; a++){
                                        if(chow_list.get(a).endsWith("11") || chow_list.get(a).endsWith("12") )
{
                                                  cl_12.add(chow_list.get(a));
                                        }
                              }
                    }

                    if(cl_12.size() == 3 || cl_13.size() == 3 || cl_23.size() == 3){
                              can_chow = 1;
                    }
          }
}
```

Deck クラス

```
package sevsen_bridge;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;


public class Deck {
          /*
           * カードの種類は　アルファベット＋数字　で表記する
           *
           * C = club
           * D = diamond
           * H = heart
           * S = spade
           */

          //デッキが０になったかどうかを判断するするときに使用する変数
```

```
public static int zero_deck = 0;

//デッキを保持するための変数
public static List<String> deck = new ArrayList<String>();

//デッキが何枚目であるかをカウントするための変数
public static int deck_num = 0;

//デッキを作成するメソッド
public static void create_deck(){
          for(int alp = 0 ; alp < 4 ; alp++){
                    for(int i = 1 ; i <= 13 ; i++){
                              if(alp == 0){
                                        if(i < 10){
                                                  deck.add("C.0" + i);
                                        }else{
                                                  deck.add("C." + i);
                                        }
                              }

                              if(alp == 1){
                                        if(i < 10){
                                                  deck.add("D.0" + i);
                                        }else{
                                                  deck.add("D." + i);
                                        }
                              }

                              if(alp == 2){
                                        if(i < 10){
                                                  deck.add("H.0" + i);
                                        }else{
                                                  deck.add("H." + i);
                                        }
                              }

                              if(alp == 3){
                                        if(i < 10){
                                                  deck.add("S.0" + i);
                                        }else{
                                                  deck.add("S." + i);
                                        }
                              }
                    }
          }
          Collections.shuffle(deck);
}
```

```
}
```

Fieldクラス

```
package sevsen_bridge;

import java.util.ArrayList;


public class Field {
	//捨札を格納するメソッド
	public static String trash_card;
	public static ArrayList<ArrayList<String>> meld_group = new ArrayList<ArrayList<String>>();
	public static ArrayList<String> meld_group_contents = new ArrayList<String>();
	public static String[][] meld_field = new String [30][15];
	//左側はメルドの属性(1:同位　2:スート)
	public static int mgc[][] = new int[30][2];
	//メルドされたグループがいくつあるかを保持する変数.
	public static int meld_group_count = 0;

	//現在のフィールドを表示するメソッド
	public static void show_Field(){
		System.out.println("¥n----------------------------------------------------");
		System.out.println("現在のフィールドの状態です.");
		System.out.println("メルドグループカウント：" + meld_group_count);

		for(int y = 0 ; y < meld_group_count  ; y++){
			System.out.print("グループ"+ y + ":");
			for(int x = 0 ; x < 15; x++){
				System.out.print(meld_field[y][x] + " ");
			}
			for(int a = 0 ; a < 2 ; a++){
				System.out.print(Field.mgc[y][a] + " ");
			}
			System.out.print("¥n");
		}
		System.out.println("----------------------------------------------------");
	}

	//メルドグループの型を調べるメソッド
	public static int getMeldType(int group_num){
		return mgc[group_num][1];
	}
}
```

Gameクラス

```
package sevsen_bridge;
```

```
import java.util.Scanner;

public class Game {
	public static int player_number = 0;
	static String player;
	static Scanner kbS = new Scanner(System.in);
	static int round_cnt = 0;
	static int game_cnt = 0;
	static int finish = 0;
	static int p_judge;
	static int c_judge;

	public static void main(String[] args) {
		while(game_cnt < 100){//合計何ゲーム行うか
			game_cnt++;
			round_cnt = 0;
			while(round_cnt < 10){//１０ラウンドを１ゲームとするため.
				finish = 0;
				round_cnt++;

				GM.Clear();

				Deck.create_deck();

				GM.deal();

				player_number = 0;

				one_game_roop: while(true){


					for(int i = 0 ; i < 3 ;i++){
						Player.showHand(i);
					}

					GM.judge();
					if(finish == 1){
						break one_game_roop;
					}
					Player.drawCard(player_number);
					Player.showHand(player_number);

					Player.evaluation();
					Player.meld_Decide();
			meld_loop: while(Player.meld_decide == 0){
						Player.meldHand();
```

```
            if(Player.meld_decide == 1){
                        break meld_loop;
            }
}

if(Game.finish == 0){
            Player.trashHand(player_number);
}

//終了判定
if(finish == 1){
            break one_game_roop;
}

p_c_roop: while(true){
            for(int i = 0 ; i < 4 ;i++){
            }

            p_judge = 0;
            c_judge = 0;

            //ポン判定→捨てる
if(player_number == 1 && P_Meld_Light.hand.size() >= 5){

            }else{
                        Pong.pong(player_number);
            }


            //GM.judge();
            if(finish == 1){
                        break one_game_roop;
            }

            //チー判定→捨てる
if(player_number == 1 && P_Meld_Light.hand.size() >= 5){

            }else{
Chow.chow(player_number , Field.trash_card);
            }

            GM.judge();
            if(finish == 1){
                        break one_game_roop;
            }

            if(p_judge == 0 && c_judge ==0){
```

```
                                                    break p_c_roop;
                                                }
                                            }

                                            Field.show_Field();


                                            GM.set_player(player_number);
                                        }
                                    }
                                    GM.game_judge();
                                }
                                GM.final_judge();
                            }
}
```

GM クラス

```
package sevsen_bridge;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class GM {
        //それぞれのプレイヤーのトータルスコアを保持する変数
        static int P1_total_score = 0;
        static int P2_total_score = 0;
        static int P3_total_score = 0;
        static int P4_total_score = 0;

        //それぞれのプレイヤーのそのラウンドのスコアを保持する変数
        static int P1_game_score = 0;
        static int P2_game_score = 0;
        static int P3_game_score = 0;
        static int P4_game_score = 0;

        //メルドされたものをフィールドに入れる前にソートを行うためのリスト
        static List<String> tmp_meld_list = new ArrayList<String>();
        //どのプレイヤーが何度何位になったかを格納する配列
        static int[][] ranking = new int [3][3];
        //順位を判定するために使用するリスト
        static List<Integer> rank_list = new ArrayList<Integer>();
        static List<Integer> tmp_rank = new ArrayList<Integer>();

        //各変数をゲームスタート時にクリアするメソッド
        public static void Clear(){
                Game.finish = 0;
                Deck.deck.clear();
```

```
		Deck.deck_num = 0;
		P_Meld_Heavy.hand.clear();
		P_Meld_Light.hand.clear();
		P_Seven_Double.hand.clear();
		Field.meld_group.clear();
		Field.meld_group_contents.clear();
		Field.meld_group_count = 0;
		for(int i = 0 ; i < 30 ; i++){
			for(int a = 0 ; a < 2 ; a++){
				Field.mgc[i][a] = 0;
			}
		}
		for(int y = 0 ; y < 30 ; y++){
			for(int x = 0 ; x < 15; x++){
				Field.meld_field[y][x] = null;
			}
		}
	}

	//各プレイヤーに手札を配るメソッド
	public static void deal() {
		for(int count = 0 ; count < 7 ; count++){
			for(int pl = 0 ; pl < 3 ; pl++){
				if(pl == 0){
					P_Meld_Heavy.hand.add(Deck.deck.get(Deck.deck_num));
					Deck.deck_num++;
				}else if(pl == 1){
					P_Meld_Light.hand.add(Deck.deck.get(Deck.deck_num));
					Deck.deck_num++;
				}else if(pl == 2){
					P_Seven_Double.hand.add(Deck.deck.get(Deck.deck_num));
					Deck.deck_num++;
				}
			}
		}
	}

	//ラウンドの終了判定を行うメソッド
	public static void judge() {
		for(int pl = 0 ; pl < 4 ; pl++){
			if(pl == 0){
				if(P_Meld_Heavy.hand.size() == 0){
					System.out.println("P1の勝利です.");
					Game.finish = 1;
				}
			}else if(pl == 1){
				if(P_Meld_Light.hand.size() == 0){
```

```
                                        System.out.println("P2の勝利です.");
                                        Game.finish = 1;
                                }
                        }else if(pl == 2){
                                if(P_Seven_Double.hand.size() == 0){
                                        System.out.println("P4の勝利です.");
                                        Game.finish = 1;
                                }
                        }
                }

                if(Deck.deck_num == 51){
                        Game.finish = 1;
                        System.out.println("デッキが無くなったので終了。");
                        Deck.zero_deck++;
                }

                if(Game.finish == 1){
                        score();
                }
        }

        //得点を加算するメソッド
        public static void score(){
                int P1_score = 0;
                int P2_score = 0;
                int P3_score = 0;
                int seven_check = 0;

                for (int i = 0 ; i < P_Meld_Heavy.hand.size() ; i++){
                        String contents = P_Meld_Heavy.hand.get(i);
                        if(contents.substring(2).equals("07")){
                                seven_check ++;
                        }
                        P1_score += string_for_int(contents);
                        if(seven_check >= 1){
                                for(int seven = 0 ; seven < seven_check ; seven++){
                                        P1_score = P1_score * 2;
                                }
                        }
                        seven_check = 0;
                }

                for (int i = 0 ; i < P_Meld_Light.hand.size() ; i++){
                        String contents = P_Meld_Light.hand.get(i);
                        P2_score += string_for_int(contents);
                        if(contents.substring(2).equals("07")){
```

```
                    seven_check ++;
                }
                P2_score += string_for_int(contents);
                if(seven_check >= 1){
                    for(int seven = 0 ; seven < seven_check ; seven++){
                        P2_score = P2_score * 2;
                    }
                }
                seven_check = 0;
            }


            for (int i = 0 ; i < P_Seven_Double.hand.size() ; i++){
                String contents = P_Seven_Double.hand.get(i);
                P3_score += string_for_int(contents);

                if(contents.substring(2).equals("07")){
                    seven_check ++;
                }
                P3_score += string_for_int(contents);
                if(seven_check >= 1){
                    for(int seven = 0 ; seven < seven_check ; seven++){
                        P3_score = P3_score * 2;
                    }
                }
                seven_check = 0;
            }

            System.out.println("P1のスコア:" + P1_score);
            P1_game_score += P1_score;
            P1_total_score += P1_score;
            System.out.println("本ゲームのP1のスコア:" + P1_game_score);
            System.out.println("トータルのP1のスコア:" + P1_total_score);

            System.out.println("P2のスコア:" + P2_score);
            P2_game_score += P2_score;
            P2_total_score += P2_score;
            System.out.println("本ゲームのP2のスコア:" + P2_game_score);
            System.out.println("トータルのP2のスコア:" + P2_total_score);

            System.out.println("P3のスコア:" + P3_score);
            P3_game_score += P3_score;
            P3_total_score += P3_score;
            System.out.println("本ゲームのP3のスコア:" + P3_game_score);
            System.out.println("トータルのP3のスコア:" + P3_total_score);
    }
```

```
//文字列を数字に変更するメソッド
public static int string_for_int(String contents){
		String str = contents.substring(2);
		int num = 0;

		if(str.equals("01")){
			num = 1;
		}
		if(str.equals("02")){
			num = 2;
		}
		if(str.equals("03")){
			num = 3;
		}
		if(str.equals("04")){
			num = 4;
		}
		if(str.equals("05")){
			num = 5;
		}
		if(str.equals("06")){
			num = 6;
		}
		if(str.equals("07")){
			num = 7;
		}
		if(str.equals("08")){
			num = 8;
		}
		if(str.equals("09")){
			num = 9;
		}
		if(str.equals("10")){
			num = 10;
		}
		if(str.equals("11")){
			num = 11;
		}
		if(str.equals("12")){
			num = 12;
		}
		if(str.equals("13")){
			num = 13;
		}
		return num;
}
```

```
	//最終結果を表示するメソッド
	public static void final_judge(){


System.out.println("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");
		System.out.println("最終結果です.");
		System.out.println("トータルのP1のスコア:" + P1_total_score);
		System.out.println("トータルのP2のスコア:" + P2_total_score);
		System.out.println("トータルのP3のスコア:" + P3_total_score);
System.out.println("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

		for(int y = 0 ; y < 3 ;y++){
			for(int x = 0 ; x < 3 ; x++){
				System.out.print(ranking[y][x] + " ");
			}
			System.out.println("");
		}
	}

	//ゲームの勝ち負けを判定するメソッド
	public static void game_judge(){
		rank_list.add(P1_game_score);
		rank_list.add(P2_game_score);
		rank_list.add(P3_game_score);

		tmp_rank.add(P1_game_score);
		tmp_rank.add(P2_game_score);
		tmp_rank.add(P3_game_score);
		Collections.sort(tmp_rank);

		System.out.println("ソート前:" + rank_list);
		System.out.println("ソート後:" + tmp_rank);

			for(int i = 0 ; i < 3 ; i++){
				if(tmp_rank.get(i) == P1_game_score){
					ranking[i][0]++;
				}
				else if(tmp_rank.get(i) == P2_game_score){
					ranking[i][1]++;
				}
				else if(tmp_rank.get(i) == P3_game_score){
					ranking[i][2]++;
				}
		}
		rank_list.clear();
		tmp_rank.clear();
		P1_game_score = 0;
```

```
                P2_game_score = 0;
                P3_game_score = 0;
        }

        //現在のプレイヤーナンバーから次のプレイヤーナンバーに変更するメソッド
        public static void set_player(int p_number) {
                if(p_number == 0){
                        Game.player_number = 1;
                }else if(p_number == 1){
                        Game.player_number = 2;
                }else if(p_number == 2){
                        Game.player_number = 0;
                }
        }

        //メルドされたカードをソートするために一時的に使用するメソッド
        public static void meld_sort(List<String> PorC_list){
                tmp_meld_list.clear();
                for(int i = 0 ; i < PorC_list.size() ; i++){
                        tmp_meld_list.add(PorC_list.get(i));
                }
                Collections.sort(tmp_meld_list);
        }
}
```

MeldRuleクラス

```
package sevsen_bridge;

import java.util.ArrayList;

public class Meld_Rule {
        public static int can_plus = 1;

        public static void choice_a_card(ArrayList<String> hand){
                Player.evaluationList.clear();
                can_plus = 1;

                for(int size = 0 ; size < hand.size() ; size++){
                        String card = hand.get(size);
                        String card_mark = card.substring(0,1);
                        String card_num = card.substring(2);

                        if(Game.player_number == 0){
                                if(card_num.equals("07")){
                                        Field.meld_field[Field.meld_group_count][0] = card;
                                        Field.mgc[Field.meld_group_count][0]++;
                                        Field.mgc[Field.meld_group_count][1] = 3;
```

```
                                        Field.meld_group_count++;
                                        remove_card(card);
                                        can_plus = 0;
                                }else{
                                        soot_plus(card);
                                }
                        }

                        if(Game.player_number == 1 && P_Meld_Light.do_meld == 1){
                                if(card_num.equals("07")){
                                        P_Meld_Light.card_cnt++;
                                        P_Meld_Light.decideList.remove(card);
                                        can_plus = 0;
                                }else{
                                        soot_plus(card);
                                }
                        }

                        if(Game.player_number == 2){
                                if(card_num.equals("07")){
                        for(int handsize = 0 ; handsize < P_Seven_Double.hand.size() ; handsize++){

if(card_mark.equals(P_Seven_Double.hand.get(handsize).substring(0,1))){

if(P_Seven_Double.hand.get(handsize).substring(2).equals("06")
                        || P_Seven_Double.hand.get(handsize).substring(2).equals("08")){

Field.meld_field[Field.meld_group_count][0] = card;
Field.mgc[Field.meld_group_count][0]++;
Field.mgc[Field.meld_group_count][1] = 3;
                                                                        Field.meld_group_count++;
                                                                        remove_card(card);
                                                                        can_plus = 0;
                                                                }
                                                        }
                                        }
                                }
                                else{
                                        soot_plus(card);
                                }
                        }

                        card = null;
                }
}

public static void soot_plus(String card){
```

```
if(card.startsWith("C")){
        for(int group = 0 ; group < 30 ; group++){
                int g_type = Field.mgc[group][1];
                if(g_type == 2 || g_type == 3){
                        if(Field.meld_field[group][0].startsWith("C")){
                                one_soot_rule(group,card,g_type);
                        }
                }
        }
}

if(card.startsWith("D")){
        for(int group = 0 ; group < 30 ; group++){
                int g_type = Field.mgc[group][1];
                if(g_type == 2 || g_type == 3){
                        if(Field.meld_field[group][0].startsWith("D")){
                                one_soot_rule(group,card,g_type);
                        }
                }
        }
}

if(card.startsWith("H")){
        for(int group = 0 ; group < 30 ; group++){
                int g_type = Field.mgc[group][1];
                if(g_type == 2 || g_type == 3){
                        if(Field.meld_field[group][0].startsWith("H")){
                                one_soot_rule(group,card,g_type);
                        }
                }
        }
}

if(card.startsWith("S")){
        for(int group = 0 ; group < 30 ; group++){
                int g_type = Field.mgc[group][1];
                if(g_type == 2 || g_type == 3){
                        if(Field.meld_field[group][0].startsWith("S")){
                                one_soot_rule(group,card,g_type);
                        }
                }
        }
}

for(int group = 0 ; group <30 ; group++){
        int g_type = Field.mgc[group][1];
```

```
                int g_size = Field.mgc[group][0];
                if(g_type == 1 || g_type == 3){
                        if(card.substring(2).equals(Field.meld_field[group][0].substring(2))){
                                ////System.out.println("同意札１枚めるどの確認：");
                                Field.meld_field[group][g_size] = card;
                                Field.mgc[group][0]++;
                                if(g_type == 3){
                                        Field.mgc[group][1] = 1;
                                }
                                remove_card(card);
                                can_plus = 0;
                        }
                }
        }
}

private static void one_soot_rule(int group,String card,int g_type) {
        int g_size = Field.mgc[group][0];

        //メルドされているカードより小さいものをメルドする
        if(Game.player_number == 1 && P_Meld_Light.do_meld == 1){
                if(GM.string_for_int(Field.meld_field[group][0])-1 == GM.string_for_int(card)){
                        for(int move = g_size ; move > 0 ; move--){
                        Field.meld_field[group][move] = Field.meld_field[group][move-1];
                        }
                        Field.meld_field[group][0] = card;
                        Field.mgc[group][0]++;
                        if(g_type == 3){
                                Field.mgc[group][1] = 2;
                        }
                }
        }

        else if(GM.string_for_int(Field.meld_field[group][0])-1 == GM.string_for_int(card)){
                for(int move = g_size ; move > 0 ; move--){
                        Field.meld_field[group][move] = Field.meld_field[group][move-1];
                }
                Field.meld_field[group][0] = card;
                Field.mgc[group][0]++;
                if(g_type == 3){
                        Field.mgc[group][1] = 2;
                }
        }

        //メルドされているカードより大きいものをメルドする
        else if(GM.string_for_int(Field.meld_field[group][g_size-1])+1 == GM.string_for_int(card)){
                Field.meld_field[group][g_size] = card;
```

```
                        Field.mgc[group][0]++;
                        if(g_type == 3){
                                Field.mgc[group][1] = 2;
                        }
                }
        }

        private static void remove_card(String card) {
                if(Game.player_number == 0){
                        while(P_Meld_Heavy.hand.remove(card)){};
                }
                if(Game.player_number == 1){
                        while(P_Meld_Light.hand.remove(card)){};
                }
                if(Game.player_number == 2){
                        while(P_Seven_Double.hand.remove(card)){};
                }
        }
}
```

P_Meld_Heavy クラス

```
package sevsen_bridge;

import java.util.ArrayList;

public class P_Meld_Heavy{
        //P_Meld_handの手札を格納するリスト
        static ArrayList<String> hand = new ArrayList<String>();
        static ArrayList<String> keepList = new ArrayList<String>();

        //メルドするカードを選択する際の状況判断に使う変数
        static int check_1st_time;
        static int tmp_check;

        //カードの選択を行うメソッド
        public static void choice_card() {
                ArrayList<String> tmpList = new ArrayList<String>();
                check_1st_time = 0;
                tmp_check = 0;

                for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).startsWith("C")){
                                tmpList.add(hand.get(i));
                        }
                }

                soot(tmpList);
```

```
            tmpList.clear();

            for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).startsWith("D")){
                                    tmpList.add(hand.get(i));
                        }
            }

            soot(tmpList);
            tmpList.clear();

            for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).startsWith("H")){
                                    tmpList.add(hand.get(i));
                        }
            }
            soot(tmpList);
            tmpList.clear();

            for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).startsWith("S")){
                                    tmpList.add(hand.get(i));
                        }
            }
            soot(tmpList);
            tmpList.clear();

            //同意札が何枚あるか数えて比較
            same_num();
}

//スート型のグループを探すメソッド
public static void soot(ArrayList<String> tmpList ){
            if(tmpList.size() != 0){
                        ArrayList<Integer> intList = new ArrayList<Integer>();
                        ArrayList<String> countList = new ArrayList<String>();
                        ArrayList<String> tmp_countList = new ArrayList<String>();
                        ArrayList<String> tmp_countList_sub = new ArrayList<String>();

                        intList.clear();
                        countList.clear();
                        tmp_countList.clear();
                        tmp_countList_sub.clear();

                        int in_Check = 0;

                        for(int i = 0 ; i < tmpList.size() ; i++){
```

```
                    intList.add(GM.string_for_int(tmpList.get(i)));
                }

                if(tmpList.size() == 1){
                    if(check_1st_time == 0){
                        Player.evaluationList = new ArrayList<String>(tmpList);
                    }
                    else if(Player.evaluationList.size() == 1 &&
GM.string_for_int(tmpList.get(0)) > GM.string_for_int(Player.evaluationList.get(0)) ){
                        Player.evaluationList = new ArrayList<String>(tmpList);
                    }
                    check_1st_time = 1;
                }else{
                    for(int count = 0 ; count < intList.size()-1 ; count++){
                        in_Check = 0;

                        if(intList.get(count)+1 == intList.get(count+1)){
                            in_Check = 1;

                            tmp_countList_sub.add(tmpList.get(count));

                        if(count + 2 > intList.size() - 1){
                            tmp_countList_sub.add(tmpList.get(count+1));
                        if(tmp_countList_sub.size() > countList.size()){
                                    countList =
                                new  ArrayList<String>(tmp_countList_sub);
                                            tmp_countList_sub.clear();
                                            tmp_check = 1;
                                    }
                                }

                        if(count + 2 < intList.size()){
                            if(intList.get(count)+2 != intList.get(count+2)){
                                tmp_countList_sub.add(tmpList.get(count+1));
                                if(tmp_countList.size() > countList.size()){
                                                    tmp_countList =
new ArrayList<String>(tmp_countList_sub);
                                                    tmp_check = 1;
                                            }
                                    }
                                }
                            }

                        if(in_Check == 0){
                            if(tmp_countList_sub.size() > countList.size()){
                                countList = new ArrayList<String>(tmp_countList_sub);
                                            tmp_check = 1;
```

```
                                                tmp_countList_sub.clear();
                                        }
                                }
                        }

                        if(tmp_check == 0){
                                countList.add(tmpList.get(tmpList.size()-1));
                        }

                        if(countList.size() == 0){
                                countList.add(tmpList.get(tmpList.size()-1));
                        }

                        if(check_1st_time == 0){
                                Player.evaluationList = new ArrayList<String>(countList);
                        }
                        else if(Player.evaluationList.size() == 0){
                                Player.evaluationList = new ArrayList<String>(countList);

                        }
                        else if(countList.size() > Player.evaluationList.size()){
                                Player.evaluationList = new ArrayList<String>(countList);
                        }
                        else if(countList.size() == Player.evaluationList.size()){
                                if(GM.string_for_int(countList.get(countList.size()-1)) >
GM.string_for_int(Player.evaluationList.get(Player.evaluationList.size()-1)) ){
                        Player.evaluationList = new ArrayList<String>(countList);
                                }
                        }
                        countList.clear();
                }
                check_1st_time = 1;
        }
}

        //同位型のグループを探すメソッド
        private static void same_num() {
                ArrayList<String> countList = new ArrayList<String>();

                for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).endsWith("01")){
                                countList.add(hand.get(i));
                        }
                }
                set_same_num(countList);

                for(int i = 0 ; i < hand.size() ; i++){
```

```
            if(hand.get(i).endsWith("02")){
                        countList.add(hand.get(i));
            }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
            if(hand.get(i).endsWith("03")){
                        countList.add(hand.get(i));
            }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
            if(hand.get(i).endsWith("04")){
                        countList.add(hand.get(i));
            }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
            if(hand.get(i).endsWith("05")){
                        countList.add(hand.get(i));
            }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
            if(hand.get(i).endsWith("06")){
                        countList.add(hand.get(i));
            }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
            if(hand.get(i).endsWith("07")){
                        countList.add(hand.get(i));
            }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
            if(hand.get(i).endsWith("01")){
                        countList.add(hand.get(i));
            }
}
set_same_num(countList);
```

```
            for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).endsWith("08")){
                                    countList.add(hand.get(i));
                        }
            }
            set_same_num(countList);

            for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).endsWith("09")){
                                    countList.add(hand.get(i));
                        }
            }
            set_same_num(countList);

            for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).endsWith("10")){
                                    countList.add(hand.get(i));
                        }
            }
            set_same_num(countList);

            for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).endsWith("11")){
                                    countList.add(hand.get(i));
                        }
            }
            set_same_num(countList);

            for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).endsWith("12")){
                                    countList.add(hand.get(i));
                        }
            }
            set_same_num(countList);

            for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).endsWith("13")){
                                    countList.add(hand.get(i));
                        }
            }
            set_same_num(countList);

}

//同位型のグループをメルド用のリストに格納するためのメソッド
private static void set_same_num(ArrayList<String> countList) {
```

```
                int list_size1 = countList.size() -1;
                int list_size2 = Player.evaluationList.size() -1;

                if(list_size1 == -1){
                        list_size1 = 0;
                }

                if(list_size2 == -1){
                        list_size2 = 0;
                }

                if(countList.size() > Player.evaluationList.size()){
                        Player.evaluationList = new ArrayList<String>(countList);
                        if(Player.evaluationList.size() == countList.size() &&
                                        GM.string_for_int(countList.get(list_size1)) >
GM.string_for_int(Player.evaluationList.get(list_size2))){
                                Player.evaluationList = new ArrayList<String>(countList);
                        }
                }
                countList.clear();
        }
}
```

P_Meld_Light クラス

```
package sevsen_bridge;

import java.util.ArrayList;

public class P_Meld_Light{
        //P_Meld_handの手札を格納するリスト
        static ArrayList<String> hand = new ArrayList<String>();
        static ArrayList<String> keepList = new ArrayList<String>();

        //メルドするカードを選択する際の状況判断に使う変数
        static int check_1st_time;
        static int tmp_check;

        static ArrayList<String> decideList = new ArrayList<String>();
        //メルド出来るカードが４枚以上あるかを数える変数
        static int card_cnt;
        //メルドするかどうかを指定するための変数 0で行う
        static int do_meld = 1;


        public static void decide_meld(){
                do_meld = 1;
```

```
		hantei: while(true){
			if(Player.evaluationList.size() >= 3){
				card_cnt += Player.evaluationList.size();
				for(int i = 0 ; i < Player.evaluationList.size() ; i++){
					while(decideList.remove(Player.evaluationList.get(i))){}
				}
			}else{
				Player.evaluationList.clear();
				break hantei;
			}
			Player.evaluationList.clear();
		}

		Meld_Rule.choice_a_card(decideList);

		if(card_cnt >= 4){
			do_meld = 0;
		}
}

//カードを選択するメソッド
public static void choice_card() {
		ArrayList<String> tmpList = new ArrayList<String>();
		check_1st_time = 0;
		tmp_check = 0;

		for(int i = 0 ; i < hand.size() ; i++){
			if(hand.get(i).startsWith("C")){
				tmpList.add(hand.get(i));
			}
		}

		soot(tmpList);
		tmpList.clear();


		for(int i = 0 ; i < hand.size() ; i++){
			if(hand.get(i).startsWith("D")){
				tmpList.add(hand.get(i));
			}
		}

		soot(tmpList);
		tmpList.clear();

		for(int i = 0 ; i < hand.size() ; i++){
```

```
                    if(hand.get(i).startsWith("H")){
                        tmpList.add(hand.get(i));
                    }
                }
                soot(tmpList);
                tmpList.clear();

                for(int i = 0 ; i < hand.size() ; i++){
                    if(hand.get(i).startsWith("S")){
                        tmpList.add(hand.get(i));
                    }
                }
                soot(tmpList);
                tmpList.clear();

                same_num();
            }

            //スート型のグループを探すメソッド
            public static void soot(ArrayList<String> tmpList ){
                if(tmpList.size() != 0){
                    ArrayList<Integer> intList = new ArrayList<Integer>();
                    ArrayList<String> countList = new ArrayList<String>();
                    ArrayList<String> tmp_countList = new ArrayList<String>();
                    ArrayList<String> tmp_countList_sub = new ArrayList<String>();

                    intList.clear();
                    countList.clear();
                    tmp_countList.clear();
                    tmp_countList_sub.clear();

                    int in_Check = 0;

                    for(int i = 0 ; i < tmpList.size() ; i++){
                        intList.add(GM.string_for_int(tmpList.get(i)));
                    }

                    if(tmpList.size() == 1){
                        if(check_1st_time == 0){
                            Player.evaluationList = new ArrayList<String>(tmpList);
                        }
                        else if(Player.evaluationList.size() == 1 &&
GM.string_for_int(tmpList.get(0)) > GM.string_for_int(Player.evaluationList.get(0)) ){
                            Player.evaluationList = new ArrayList<String>(tmpList);
                        }
                        check_1st_time = 1;
                    }else{
```

```
for(int count = 0 ; count < intList.size()-1 ; count++){
            in_Check = 0;

            if(intList.get(count)+1 == intList.get(count+1)){
                        in_Check = 1;

                        tmp_countList_sub.add(tmpList.get(count));

                        if(count + 2 > intList.size() - 1){

        tmp_countList_sub.add(tmpList.get(count+1));
                                    if(tmp_countList_sub.size() >
countList.size()){
                                                countList = new
ArrayList<String>(tmp_countList_sub);
                                                tmp_countList_sub.clear();
                                                tmp_check = 1;
                                    }
                        }

            if(count + 2 < intList.size()){
            if(intList.get(count)+2 != intList.get(count+2)){
                        tmp_countList_sub.add(tmpList.get(count+1));
            if(tmp_countList.size() > countList.size()){
                        tmp_countList = new ArrayList<String>(tmp_countList_sub);
                                                tmp_check = 1;
                                    }
                        }
                        }
                        }

                        if(in_Check == 0){
            if(tmp_countList_sub.size() > countList.size()){
                        countList = new ArrayList<String>(tmp_countList_sub);
                                    tmp_check = 1;
                                    tmp_countList_sub.clear();
                        }
                        }
            }

            if(tmp_check == 0){
                        countList.add(tmpList.get(tmpList.size()-1));
            }

            if(countList.size() == 0){
                        countList.add(tmpList.get(tmpList.size()-1));
            }
```

```
                                        if(check_1st_time == 0){
                                                Player.evaluationList = new ArrayList<String>(countList);
                                        }
                                        else if(Player.evaluationList.size() == 0){
                                                Player.evaluationList = new ArrayList<String>(countList);

                                        }
                                        else if(countList.size() > Player.evaluationList.size()){
                                                Player.evaluationList = new ArrayList<String>(countList);
                                        }
                                        else if(countList.size() == Player.evaluationList.size()){
                                        if(GM.string_for_int(countList.get(countList.size()-1)) >
GM.string_for_int(Player.evaluationList.get(Player.evaluationList.size()-1)) ){
                                        Player.evaluationList = new ArrayList<String>(countList);
                                                }
                                        }
                                        countList.clear();
                                }
                                check_1st_time = 1;
                        }

                }

                //同位型のグループを探すメソッド
                private static void same_num() {
                        ArrayList<String> countList = new ArrayList<String>();

                        for(int i = 0 ; i < hand.size() ; i++){
                                if(hand.get(i).endsWith("01")){
                                        countList.add(hand.get(i));
                                }
                        }
                        set_same_num(countList);

                        for(int i = 0 ; i < hand.size() ; i++){
                                if(hand.get(i).endsWith("02")){
                                        countList.add(hand.get(i));
                                }
                        }
                        set_same_num(countList);

                        for(int i = 0 ; i < hand.size() ; i++){
                                if(hand.get(i).endsWith("03")){
                                        countList.add(hand.get(i));
                                }
                        }
```

```
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("04")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("05")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("06")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("07")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("01")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("08")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("09")){
                    countList.add(hand.get(i));
```

```
                }
        }
        set_same_num(countList);

        for(int i = 0 ; i < hand.size() ; i++){
                if(hand.get(i).endsWith("10")){
                        countList.add(hand.get(i));
                }
        }
        set_same_num(countList);

        for(int i = 0 ; i < hand.size() ; i++){
                if(hand.get(i).endsWith("11")){
                        countList.add(hand.get(i));
                }
        }
        set_same_num(countList);

        for(int i = 0 ; i < hand.size() ; i++){
                if(hand.get(i).endsWith("12")){
                        countList.add(hand.get(i));
                }
        }
        set_same_num(countList);

        for(int i = 0 ; i < hand.size() ; i++){
                if(hand.get(i).endsWith("13")){
                        countList.add(hand.get(i));
                }
        }
        set_same_num(countList);

}

//同位型のグループをメルド用のリストに格納するためのメソッド
private static void set_same_num(ArrayList<String> countList) {
        int list_size1 = countList.size() -1;
        int list_size2 = Player.evaluationList.size() -1;

        if(list_size1 == -1){
                list_size1 = 0;
        }

        if(list_size2 == -1){
                list_size2 = 0;
        }
```

```
                    if(countList.size() > Player.evaluationList.size()){
                              Player.evaluationList = new ArrayList<String>(countList);
                              if(Player.evaluationList.size() == countList.size() &&
                                                  GM.string_for_int(countList.get(list_size1)) >
GM.string_for_int(Player.evaluationList.get(list_size2))){
                                        Player.evaluationList = new ArrayList<String>(countList);
                              }
                    }
                    countList.clear();
          }
}
```

P_Seven_Double クラス

```
package sevsen_bridge;

import java.util.ArrayList;

public class P_Seven_Double {
          //P_Seven_Doubleの手札を格納するリスト
          static ArrayList<String> hand = new ArrayList<String>();
          static ArrayList<String> keepList = new ArrayList<String>();

          //メルドするカードを選択する際の状況判断に使う変数
          static int check_1st_time;
          static int tmp_check;

          //カードの選択を行うメソッド
          public static void choice_card() {
                    ArrayList<String> tmpList = new ArrayList<String>();
                    check_1st_time = 0;
                    tmp_check = 0;

                    for(int i = 0 ; i < hand.size() ; i++){
                              if(hand.get(i).startsWith("C")){
                                        tmpList.add(hand.get(i));
                              }
                    }

                    soot(tmpList);
                    tmpList.clear();

                    for(int i = 0 ; i < hand.size() ; i++){
                              if(hand.get(i).startsWith("D")){
                                        tmpList.add(hand.get(i));
                              }
                    }
```

```
                soot(tmpList);
                tmpList.clear();

                for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).startsWith("H")){
                                tmpList.add(hand.get(i));
                        }
                }
                soot(tmpList);
                tmpList.clear();

                for(int i = 0 ; i < hand.size() ; i++){
                        if(hand.get(i).startsWith("S")){
                                tmpList.add(hand.get(i));
                        }
                }
                soot(tmpList);
                tmpList.clear();

                same_num();
        }

        //スート型のグループを探すメソッド
        public static void soot(ArrayList<String> tmpList ){
                if(tmpList.size() != 0){
                        ArrayList<Integer> intList = new ArrayList<Integer>();
                        ArrayList<String> countList = new ArrayList<String>();
                        ArrayList<String> tmp_countList = new ArrayList<String>();
                        ArrayList<String> tmp_countList_sub = new ArrayList<String>();

                        intList.clear();
                        countList.clear();
                        tmp_countList.clear();
                        tmp_countList_sub.clear();

                        int in_Check = 0;

                        for(int i = 0 ; i < tmpList.size() ; i++){
                                intList.add(GM.string_for_int(tmpList.get(i)));
                        }

                        if(tmpList.size() == 1){
                                if(check_1st_time == 0){
                                        Player.evaluationList = new ArrayList<String>(tmpList);
                                }
        else if(Player.evaluationList.size() == 1 && GM.string_for_int(tmpList.get(0)) >
GM.string_for_int(Player.evaluationList.get(0)) ){
```

```
                    Player.evaluationList = new ArrayList<String>(tmpList);
                }
                check_1st_time = 1;
            }else{
                for(int count = 0 ; count < intList.size()-1 ; count++){
                    in_Check = 0;

                    if(intList.get(count)+1 == intList.get(count+1)){
                        in_Check = 1;

                        tmp_countList_sub.add(tmpList.get(count));

                        if(count + 2 > intList.size() - 1){
                        tmp_countList_sub.add(tmpList.get(count+1));
                        if(tmp_countList_sub.size() > countList.size()){
                    countList = new ArrayList<String>(tmp_countList_sub);
                                        tmp_countList_sub.clear();
                                        tmp_check = 1;
                                }
                        }

                        if(count + 2 < intList.size()){
                    if(intList.get(count)+2 != intList.get(count+2)){
                        tmp_countList_sub.add(tmpList.get(count+1));
                    if(tmp_countList.size() > countList.size()){
                    tmp_countList = new ArrayList<String>(tmp_countList_sub);
                                                tmp_check = 1;
                                        }
                                }
                        }
                    }

                    if(in_Check == 0){
                        if(tmp_countList_sub.size() > countList.size()){
                    countList = new ArrayList<String>(tmp_countList_sub);
                                tmp_check = 1;
                                tmp_countList_sub.clear();
                        }
                    }
                }

                if(tmp_check == 0){
                    countList.add(tmpList.get(tmpList.size()-1));
                }

                if(countList.size() == 0){
                    countList.add(tmpList.get(tmpList.size()-1));
```

```
                                        }

                                        if(check_1st_time == 0){
                                                Player.evaluationList = new ArrayList<String>(countList);
                                        }
                                        else if(Player.evaluationList.size() == 0){
                                                Player.evaluationList = new ArrayList<String>(countList);

                                        }
                                        else if(countList.size() > Player.evaluationList.size()){
                                                Player.evaluationList = new ArrayList<String>(countList);
                                        }
                                        else if(countList.size() == Player.evaluationList.size()){
                                                if(GM.string_for_int(countList.get(countList.size()-1)) >
GM.string_for_int(Player.evaluationList.get(Player.evaluationList.size()-1)) ){
                                        Player.evaluationList = new ArrayList<String>(countList);
                                                }
                                        }
                                        countList.clear();
                                }
                                check_1st_time = 1;
                        }
                }

                //同位型のグループを探すメソッド
                private static void same_num() {
                        ArrayList<String> countList = new ArrayList<String>();

                        for(int i = 0 ; i < hand.size() ; i++){
                                if(hand.get(i).endsWith("01")){
                                        countList.add(hand.get(i));
                                }
                        }
                        set_same_num(countList);

                        for(int i = 0 ; i < hand.size() ; i++){
                                if(hand.get(i).endsWith("02")){
                                        countList.add(hand.get(i));
                                }
                        }
                        set_same_num(countList);

                        for(int i = 0 ; i < hand.size() ; i++){
                                if(hand.get(i).endsWith("03")){
                                        countList.add(hand.get(i));
                                }
                        }
```

```
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("04")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("05")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("06")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("07")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("01")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("08")){
                    countList.add(hand.get(i));
          }
}
set_same_num(countList);

for(int i = 0 ; i < hand.size() ; i++){
          if(hand.get(i).endsWith("09")){
                    countList.add(hand.get(i));
```

```
                }
            }
            set_same_num(countList);

            for(int i = 0 ; i < hand.size() ; i++){
                if(hand.get(i).endsWith("10")){
                        countList.add(hand.get(i));
                }
            }
            set_same_num(countList);

            for(int i = 0 ; i < hand.size() ; i++){
                if(hand.get(i).endsWith("11")){
                        countList.add(hand.get(i));
                }
            }
            set_same_num(countList);

            for(int i = 0 ; i < hand.size() ; i++){
                if(hand.get(i).endsWith("12")){
                        countList.add(hand.get(i));
                }
            }
            set_same_num(countList);

            for(int i = 0 ; i < hand.size() ; i++){
                if(hand.get(i).endsWith("13")){
                        countList.add(hand.get(i));
                }
            }
            set_same_num(countList);

    }

    //同位型のグループをメルド用のリストに格納するためのメソッド
    private static void set_same_num(ArrayList<String> countList) {
            int list_size1 = countList.size() -1;
            int list_size2 = Player.evaluationList.size() -1;

            if(list_size1 == -1){
                list_size1 = 0;
            }

            if(list_size2 == -1){
                list_size2 = 0;
            }
```

```
                    if(countList.size() > Player.evaluationList.size()){
                              Player.evaluationList = new ArrayList<String>(countList);
                              if(Player.evaluationList.size() == countList.size() &&
                                                  GM.string_for_int(countList.get(list_size1)) >
GM.string_for_int(Player.evaluationList.get(list_size2))){
                                        Player.evaluationList = new ArrayList<String>(countList);
                              }
                    }
                    countList.clear();
          }
}
```

Player クラス

```
package sevsen_bridge;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class Player {

          /*
           * 作戦の一覧
           * Meld_Heavy : 積極的にメルドをする(１枚,一組でも出せるようになったら行う)
           * Meld_Light : メルドをあまりしない（手札を４枚以上出せるまで)
           * High_Double : 7は２枚以上で出せるようになるまで出さない
           */
          static Scanner kbS = new Scanner(System.in);
          static ArrayList<String> evaluationList = new ArrayList<String>();
          static ArrayList<String> soot = new ArrayList<String>();
          public static int max = 0;
          public static int index_num = 0;
          public static int meld_decide = 0;
          static int can_meld = 0;
          public static int type = 0;


          //デッキからプレーヤーにカードを配る
          public static void drawCard(int player_number) {
                    if(player_number == 0){
                              P_Meld_Heavy.hand.add(Deck.deck.get(Deck.deck_num));
                              Deck.deck_num++;
                    }else if(player_number == 1){
                              P_Meld_Light.hand.add(Deck.deck.get(Deck.deck_num));
                              Deck.deck_num++;
                    }else if(player_number == 2){
```

```
                P_Seven_Double.hand.add(Deck.deck.get(Deck.deck_num));
                Deck.deck_num++;
        }
}

//手札を表示する
public static void showHand(int player_number){
        if(player_number == 0){
                System.out.println("MH:");
                Collections.sort(P_Meld_Heavy.hand);
                for (int i = 0 ; i < P_Meld_Heavy.hand.size() ; i++){
                        String contents = P_Meld_Heavy.hand.get(i);
                        System.out.print(contents);
                        System.out.print("  ");
                }
                System.out.print("¥n");
        }
        else if(player_number == 1){
                System.out.println("ML:");
                Collections.sort(P_Meld_Light.hand);
                for (int i = 0 ; i < P_Meld_Light.hand.size() ; i++){
                        String contents = P_Meld_Light.hand.get(i);
                        System.out.print(contents);
                        System.out.print("  ");
                }
                System.out.print("¥n");
        }
        else if(player_number == 2){
                System.out.println("HD:");
                Collections.sort(P_Seven_Double.hand);
                for (int i = 0 ; i < P_Seven_Double.hand.size() ; i++){
                        String contents = P_Seven_Double.hand.get(i);
                        System.out.print(contents);
                        System.out.print("  ");
                }
                System.out.print("¥n");
        }
}

//手札を捨てるメソッド
public static void trashHand(int player_number){

        if(player_number == 0){
                choiceTrashHand(P_Meld_Heavy.hand);
                for (int i = 0 ; i < P_Meld_Heavy.hand.size() ; i++){
                        if(Field.trash_card.equals(P_Meld_Heavy.hand.get(i))){
                                P_Meld_Heavy.hand.remove(i);
```

```
                    }
                }
            }else if(player_number == 1){
                choiceTrashHand(P_Meld_Light.hand);
                for (int i = 0 ; i < P_Meld_Light.hand.size() ; i++){
                    if(Field.trash_card.equals(P_Meld_Light.hand.get(i))){
                        P_Meld_Light.hand.remove(i);
                    }
                }
            }else if(player_number == 2){
                choiceTrashHand(P_Seven_Double.hand);
                for (int i = 0 ; i < P_Seven_Double.hand.size() ; i++){
                    if(Field.trash_card.equals(P_Seven_Double.hand.get(i))){
                        P_Seven_Double.hand.remove(i);
                    }
                }
            }
            GM.judge();
    }

    //メルドをするかどうか決定するメソッド
    public static void meld_Decide() {
            if(Player.evaluationList.size() >= 3){//現在３枚以上で出せる分だけなので（付け足せない）
                meld_decide = 0;
            }else if(Player.evaluationList.size() < 3){
                can_meld = 1;
                do_choice_a_card();
                if(Meld_Rule.can_plus == 0){
                    meld_decide = 0;
                }else if(Meld_Rule.can_plus == 1 && can_meld == 1){
                    meld_decide = 1;
                }
            }
    }

    //メルドするカードを１枚選ぶ時のメソッド
    private static void do_choice_a_card() {
            if(Game.player_number == 0){
                Meld_Rule.choice_a_card(P_Meld_Heavy.hand);
            }
            if(Game.player_number == 1){
                Meld_Rule.choice_a_card(P_Meld_Light.hand);
            }
            if(Game.player_number == 2){
                Meld_Rule.choice_a_card(P_Seven_Double.hand);
            }
    }
```

```
//有効手を格納するメソッド
public static void evaluation(){

		if(Game.player_number == 0){
			P_Meld_Heavy.choice_card();
		}

		if(Game.player_number == 1){
			P_Meld_Light.decideList = new ArrayList<String>(P_Meld_Light.hand);

			if(P_Meld_Light.do_meld == 1){
				P_Meld_Light.decide_meld();
			}else{
				P_Meld_Light.choice_card();
			}
		}

		if(Game.player_number == 2){
			P_Seven_Double.choice_card();
		}

}

//手札のカードをフィールドにメルドする時のメソッド
public static void meldHand() {
		type_check();

		if(Game.player_number == 0){
			for(int size = 0 ; size < evaluationList.size() ; size++){
			Field.meld_field [Field.meld_group_count][size] = evaluationList.get(size);
				if(Player.evaluationList.size() >= 3){
					Field.mgc[Field.meld_group_count][1] = type;
					Field.mgc[Field.meld_group_count][0]++;
				}
				while(P_Meld_Heavy.hand.remove(evaluationList.get(size))){};
			}
		}

		if(Game.player_number == 1){
			for(int size = 0 ; size < evaluationList.size() ; size++){
			Field.meld_field [Field.meld_group_count][size] = evaluationList.get(size);
				if(Player.evaluationList.size() >= 3){
					Field.mgc[Field.meld_group_count][1] = type;
					Field.mgc[Field.meld_group_count][0]++;
				}
				while(P_Meld_Light.hand.remove(evaluationList.get(size))){};
```

```
                }
        }

        if(Game.player_number == 2){
                for(int size = 0 ; size < evaluationList.size() ; size++){
                Field.meld_field [Field.meld_group_count][size] = evaluationList.get(size);
                        if(Player.evaluationList.size() >= 3){
                                Field.mgc[Field.meld_group_count][1] = type;
                                Field.mgc[Field.meld_group_count][0]++;
                        }
                        while(P_Seven_Double.hand.remove(evaluationList.get(size))){};
                }
        }

        if(Player.evaluationList.size() >= 3){
                Field.meld_group_count++;
        }
        evaluationList.clear();

        Player.evaluation();
        Player.meld_Decide();

        GM.judge();
}

//有効手を格納しているリストの型をチェックするためのメソッド
private static void type_check() {
        if(Player.evaluationList.size() >= 3){

if(Player.evaluationList.get(0).substring(2).equals(Player.evaluationList.get(1).substring(2))){
                        type = 1;
                }else{
                        type = 2;
                }
        }
}

//捨札を選択するためのメソッド
private static void choiceTrashHand(ArrayList<String> hand) {
        ArrayList<String> choice_trash = new ArrayList<String>(hand);
        ArrayList<String> same_num = new ArrayList<String>();
        Field.trash_card = null;
        soot.clear();

        //同位札を探して削除
        for(int hand_num = 0 ; hand_num < choice_trash.size() ; hand_num++){
                String str = choice_trash.get(hand_num).substring(2);
```

```
            for(int i = 0 ; i < choice_trash.size() ; i++){
                        //同じ数字のカードのがあればリストに入れる
                        if(hand_num != i){
                                    if(choice_trash.get(i).endsWith(str)){
                                                same_num.add(choice_trash.get(i));
                                    }
                        }
            }
}

ArrayList<String> sootC = new ArrayList<String>();
ArrayList<String> sootD = new ArrayList<String>();
ArrayList<String> sootH = new ArrayList<String>();
ArrayList<String> sootS = new ArrayList<String>();

sootC.clear();
sootD.clear();
sootS.clear();
sootH.clear();

//スートを探して削除していく
//それぞれの文字をリストに入れる.
for(int hand_num = 0 ; hand_num < choice_trash.size() ; hand_num++){
            if(choice_trash.get(hand_num).startsWith("C")){
                        sootC.add(choice_trash.get(hand_num));
            }

            if(choice_trash.get(hand_num).startsWith("D")){
                        sootD.add(choice_trash.get(hand_num));
            }

            if(choice_trash.get(hand_num).startsWith("H")){
                        sootH.add(choice_trash.get(hand_num));
            }

            if(choice_trash.get(hand_num).startsWith("S")){
                        sootS.add(choice_trash.get(hand_num));
            }
}

sootRule(sootC);
sootRule(sootD);
sootRule(sootH);
sootRule(sootS);

ArrayList<String> choice_trash_copy = new ArrayList<String>(choice_trash);
```

```
for(int i = 0 ; i < same_num.size() ; i++){
          for(int a = 0 ; a < choice_trash.size() ; a++){
                    if(i <= choice_trash.size()-1 && choice_trash.size() != 0){
                              choice_trash.remove(same_num.get(i));
                    }
          }
}

for(int i = 0 ; i < soot.size() ; i++){
          for(int a = 0 ; a < choice_trash.size() ; a++){
                    if(i <= choice_trash.size()-1 && choice_trash.size() != 0){
                              choice_trash.remove(soot.get(i));
                    }
          }
}

if(choice_trash.size() == 0){
          int tmp = 0;
          int index = 0;
          String str;

          for(int size = 0 ; size < choice_trash_copy.size() ; size++){
                    int num = GM.string_for_int(choice_trash_copy.get(size));
                    if(tmp > num){
                              tmp = num;
                              index = size;
                    }
          }
          str = choice_trash_copy.get(index);
          choice_trash.clear();
          choice_trash.add(str);
}

String t_card;
ArrayList<Integer> int_t = new ArrayList<Integer>();

for(int size = 0 ; size < choice_trash.size() ; size++){
          int_t.add(GM.string_for_int(choice_trash.get(size)));
}

max = 0;
index_num = 0;


for(int size = 0 ; size < int_t.size() ; size++){
          if(int_t.get(size) > max){
```

```
                                        max = int_t.get(size);
                                        index_num = size;
                                }
                        }

                        t_card = choice_trash.get(index_num);
                        Field.trash_card = t_card;
                }

                private static void sootRule(ArrayList<String> list) {
                        for(int i = 0 ; i < list.size() ; i++){
                                String num = list.get(i).substring(2);
                                if(num.equals("01")){
                                        for (int size = 0 ; size < list.size() ; size++){
                                                if(list.get(size).endsWith("02")){
                                                        soot.add(list.get(size));
                                                }
                                        }
                                }
                                if(num.equals("02")){
                                        for (int size = 0 ; size < list.size() ; size++){
                                                if(list.get(size).endsWith("01") ||
list.get(size).endsWith("03")){
                                                        soot.add(list.get(size));
                                                }
                                        }
                                }
                                if(num.equals("03")){
                                        for (int size = 0 ; size < list.size() ; size++){
                                                if(list.get(size).endsWith("02") ||
list.get(size).endsWith("04")){
                                                        soot.add(list.get(size));
                                                }
                                        }
                                }
                                if(num.equals("04")){
                                        for (int size = 0 ; size < list.size() ; size++){
                                                if(list.get(size).endsWith("03") ||
list.get(size).endsWith("05")){
                                                        soot.add(list.get(size));
                                                }
                                        }
                                }
                                if(num.equals("05")){
                                        for (int size = 0 ; size < list.size() ; size++){
                                                if(list.get(size).endsWith("04") ||
list.get(size).endsWith("06")){
```

```
                                        soot.add(list.get(size));
                                }
                        }
                }
                if(num.equals("06")){
                        for (int size = 0 ; size < list.size() ; size++){
                                if(list.get(size).endsWith("05") ||
list.get(size).endsWith("07")){
                                        soot.add(list.get(size));
                                }
                        }
                }
                if(num.equals("07")){
                        for (int size = 0 ; size < list.size() ; size++){
                        if(list.get(size).endsWith("06") || list.get(size).endsWith("08")){
                                        soot.add(list.get(size));
                                }
                        }
                }
                if(num.equals("08")){
                        for (int size = 0 ; size < list.size() ; size++){
                        if(list.get(size).endsWith("07") || list.get(size).endsWith("09")){
                                        soot.add(list.get(size));
                                }
                        }
                }
                if(num.equals("09")){
                        for (int size = 0 ; size < list.size() ; size++){
                        if(list.get(size).endsWith("08") || list.get(size).endsWith("10")){
                                        soot.add(list.get(size));
                                }
                        }
                }
                if(num.equals("10")){
                        for (int size = 0 ; size < list.size() ; size++){
                        if(list.get(size).endsWith("09") || list.get(size).endsWith("11")){
                                        soot.add(list.get(size));
                                }
                        }
                }
                if(num.equals("11")){
                        for (int size = 0 ; size < list.size() ; size++){
                                if(list.get(size).endsWith("10") ||
list.get(size).endsWith("12")){
                                        soot.add(list.get(size));
                                }
                        }
```

```
                                }
                                if(num.equals("12")){
                                        for (int size = 0 ; size < list.size() ; size++){
                                                if(list.get(size).endsWith("11") ||
list.get(size).endsWith("13")){
                                                        soot.add(list.get(size));
                                                }
                                        }
                                }
                                if(num.equals("13")){
                                        for (int size = 0 ; size < list.size() ; size++){
                                                if(list.get(size).endsWith("12")){
                                                        soot.add(list.get(size));
                                                }
                                        }
                                }
                        }
                }
}
```

Pongクラス

```
package sevsen_bridge;

import java.util.ArrayList;
import java.util.List;

public class Pong {
        public static void pong(int p_number) {
        List<String> pong_list = new ArrayList<String>();//捨てられたカードと同位の札を格納するためのリスト
                String str = Field.trash_card.substring(2,4);//文字列の3.4文字目を取り出す.

                for(int i = 0 ; i < 4 ; i++ ){
                        if(i != p_number){
                                if(i == 0){
                                        pong_list.clear();
                                        pong_list.add(Field.trash_card);
                                        for (int a = 0 ; a < P_Meld_Heavy.hand.size() ; a++){
                                                if(P_Meld_Heavy.hand.get(a).endsWith(str)){
                                                pong_list.add(P_Meld_Heavy.hand.get(a));
                                                        if(pong_list.size() >= 3){
                                                                int input_int = 1;
                                                                if(input_int == 1){
                                                                        Game.p_judge = 1;


        GM.meld_sort(pong_list);
                                                                                for(int pong = 0 ;
```

```
pong < pong_list.size() ; pong++){

            Field.meld_field [Field.meld_group_count][pong] = GM.tmp_meld_list.get(pong);

            Field.mgc[Field.meld_group_count][0]++;

            Field.mgc[Field.meld_group_count][1] = 1;

            while(P_Meld_Heavy.hand.remove(pong_list.get(pong))){};
                                                                                    }
                                                                                    Field.trash_card  =
"null";

            Field.meld_group_count++;

            if(P_Meld_Heavy.hand.size() != 0){

            Player.trashHand(0);
                                                                                    }
                                                                                    Game.player_number
= 0;
                                                                            }
                                                                    }
                                                            }
                                                    }
                                            }

                                            else if(i == 1){
                                                    pong_list.clear();
                                                    pong_list.add(Field.trash_card);
                                                    for (int a = 0 ; a < P_Meld_Light.hand.size() ; a++){
                                            if(P_Meld_Light.hand.get(a).endsWith(str)){
                                                    pong_list.add(P_Meld_Light.hand.get(a));
                                                    if(pong_list.size() >= 3){
                                                            int input_int =  0;
                    if(P_Meld_Light.hand.size() <= 4 || P_Meld_Light.hand.size() - pong_list.size() <= 4){
                                                                                    input_int = 1;
                                                                                            }
            if(input_int == 1){
            Game.p_judge = 1;
            GM.meld_sort(pong_list);
            for(int pong = 0 ; pong < pong_list.size() ; pong++){

            Field.meld_field [Field.meld_group_count][pong] = GM.tmp_meld_list.get(pong);

            Field.mgc[Field.meld_group_count][0]++;
```

```
Field.mgc[Field.meld_group_count][1] = 1;

while(P_Meld_Light.hand.remove(pong_list.get(pong))){};
                                                                                    }

Field.meld_group.add(Field.meld_group_contents);
          Field.trash_card = "null";

Field.meld_group_count++;

if(P_Meld_Light.hand.size() != 0){

Player.trashHand(1);
                                                                                    }
                                                    Game.player_number = 1;
                                                                        }
                                                              }
                                                    }
                                          }
                                }

                                else if(i == 2){
                                          pong_list.clear();
                                          pong_list.add(Field.trash_card);
                                          for (int a = 0 ; a < P_Seven_Double.hand.size() ; a++){
                                                    if(P_Seven_Double.hand.get(a).endsWith(str)){
                                                    pong_list.add(P_Seven_Double.hand.get(a));
                                                              if(pong_list.size() >= 3){
                                                                        int input_int =  1;
                                                                        if(input_int == 1){
                                                              Game.p_judge = 1;
                                                              GM.meld_sort(pong_list);
                                          for(int pong = 0 ; pong < pong_list.size() ; pong++){

Field.meld_field [Field.meld_group_count][pong] = GM.tmp_meld_list.get(pong);

Field.mgc[Field.meld_group_count][0]++;

Field.mgc[Field.meld_group_count][1] = 1;

while(P_Seven_Double.hand.remove(pong_list.get(pong))){};
                                                                                    }

Field.meld_group.add(Field.meld_group_contents);
                                                    Field.trash_card = "null";

Field.meld_group_count++;
```

```
            if(P_Seven_Double.hand.size() != 0){

            Player.trashHand(3);
                                                                                    }
                                                                Game.player_number = 2;
                                                                            }
                                                                    }
                                                                }
                                                        }
                                                }
                                        }
                                }
            }
}
```