卒業研究報告書

題目

セブンティーンポーカーにおける強いAI開発

指導教員

石水 隆　講師

報告者

17-1-037-0147

鈴木 拓磨

近畿大学理工学部情報学科

令和03年2月1日提出

概要

カードゲームの中でもポーカーは世界中でプレイされており，スタッドポーカーやドローポーカーなど様々なヴァリエーションがある．その一つがセブンティーンポーカーである[1]．一般的に知られているドローポーカーは 52 枚のカードを用いるが，セブンティーンポーカーではエースと絵札合わせて 16 枚とジョーカー1 枚で行う．ドローポーカーと同じくセブンティーンポーカーは強い役を狙って手札を交換するのが基本的な戦略であるが，カードの枚数が少なく，山札や相手のカードが予測しやすく役が揃えやすいので，ドローポーカーとは違った戦略が必要である．そこで本研究ではセブンティーンポーカーの AI を開発する．

目次

1. 序論

1.1 ポーカー

カードゲームの中でもポーカーは世界中でプレイされており，スタッドポーカーやドローポーカーなど様々なヴァリエーションがある．基本的なポーカーは 52 枚のカードを用い，対戦人数は 2〜6 人である．まずプレイヤーは場代を支払い，カードを 5 枚ずつ貰う．次に一人ずつ順に賭け金設定やゲームを降りるなどの選択を全員の賭け金が揃うまで行う．次により強い役を揃えるために手札交換をする．役は同じ数字を揃える，またはスートを揃えることによってできるもののことである．交換後は先にも述べた選択を行い，最後に全員の役を公開して役が最も強いプレイヤーが場にある賭け金を獲得する．

このルールの大きな特徴は二つある．一つはディーラーとプレイヤーの対戦ではなく，プレイヤー同士の対戦ということである．強い役ができても場にある賭け金が少ないと勝った時の儲けも少ないので，相手により多くの賭け金を出させる必要がある．もう一つは役が弱い，または揃わない場合でもゲームに勝つことができるところである．賭け金を大きく吊り上げると相手に強い役ができていると思わせ，ゲームを降りさせること(ブラフ)ができるからである．これらの特徴からポーカーは，強い役を揃えるための手札交換の仕方や運よりも，チップの賭け方で相手の心を揺さぶる力や相手の選択で強い手札かどうかを読むという心理的な要素が重要である．

1.2 ポーカーの戦略

ポーカーの基本的な戦略についてまずは手札交換の仕方を説明する．表 1 にポーカーにおける役と成立する確率を示す．この表の確率からほとんどの場合ワンペアかノーペアになるので，カードが配られた時点でその他の役が揃った場合かなり優位なことが分かる．ノーペアの場合は最も強い数字を残してその他を交換する．ワンペア，ツーペア，スリーカード，フォーカードの場合は揃っているペアを残してその他を交換する．フラッシュ，ストレートは手札交換をしてしまうと役が弱くなってしまう可能性があるので交換しない方が良い．次に行動選択について説明する．ワンペア，ノーペアの場合は基本的にゲームから降りることを選択し，失うチップを抑える．その他の役の場合は相手の行動の仕方を見ながら勝負する．相手の賭け金の大きさで役の強さを予測し，勝てると考えた場合は勝負し，負けると考えた場合はゲームから降りる．大きな負けをできるだけ回避することが基本的な戦略となっている．

表 1　ポーカーの役と成立する確率

役	役の説明	確率
ストレートフラッシュ	同じスートかつ数字が連続して揃う	0.0015%
フォーカード	同じ数字のカードが 4 枚揃う	0.024%
フルハウス	同じ数字のカードが 3 枚と同じ数字のカードが 2 枚揃う	0.14%
フラッシュ	同じスートのカードが 5 枚揃う	0.20%
ストレート	数字が連続して揃う	0.39%
スリーカード	同じ数字のカードが 3 枚揃う	2.1%
ツーペア	同じ数字のカード 2 枚組が 2 組揃う	4.75%
ワンペア	同じ数字のカードが 2 枚揃う	42.25%
ノーペア	上記の役が何も揃わない	50%

ポーカーのヴァリエーションの一つとして，ジョーカーを用いるワイルドポーカーがある．ジョー

カーは任意のカードの代わりとすることができ，手札にジョーカーがあると役を作りやすい．また，ジョーカーと同位札 4 枚が手札に揃うと，ファイブカードという役になる．表 2 に，ジョーカーを用いる場合の各役が成立する確率を示す．

表 2　ジョーカーを用いる場合の各役が成立する確率[6]

役	役の説明	確率
ファイブカード	同じ数字のカードが 5 枚揃う	0.0025%
ロイヤルフラッシュ	10,J,Q,K,A が揃う	0.0027%
ストレートフラッシュ	同じスートかつ数字が連続して揃う	0.0152%
フォーカード	同じ数字のカードが 4 枚揃う	0.296%
フルハウス	同じ数字のカードが 3 枚と同じ数字のカードが 2 枚揃う	0.296%
フラッシュ	同じスートのカードが 5 枚揃う	0.362%
ストレート	数字が連続して揃う	0.9657%
スリーカード	同じ数字のカードが 3 枚揃う	7.386%
ツーペア	同じ数字のカード 2 枚組が 2 組揃う	3.9068%
ワンペア	同じ数字のカードが 2 枚揃う	45.5481%
ノーペア	上記の役が何も揃わない	41.2192%

ジョーカーがある場合，ツーペアよりもスリーカードの方が成立しやすくなる．また，フォーカードとフルハウスが同じ確率で現れる．このような，役の強弱と成立確率が入れ替わる現象は Wild Card Poker Paradox と呼ばれる．このパラドクスがあるため，ジョーカーがある場合はスリーカードを狙っていくと効率が良い．

1.3 既存のポーカーAI

ポーカーは不確定非完全情報ゲームに分類される．不確定とはダイスを振ったり山札から引いたりすることにより，同じ着手でも結果にランダム性があることであり，非完全情報とは他のプレイヤーの手札などゲームに関する情報が分からないことである．不確定非完全情報ゲームは良い手の判定が難しく，将棋や囲碁等の確定完全情報ゲームと比べると，強いプログラムの作成は困難である．加えてポーカーは心理戦の要素もあり，それが一層ポーカーの作成を困難にしている．

しかし昨今では，不確定非完全情報ゲームに対する AI 開発も進んでおり，ポーカーAI も開発されている．2017 年にはカーネギーメロン大学が開発したポーカーAI が 4 人のプロプレイヤーを相手にテキサス・ホールデム大会で勝利した[2][3]．また,2019 年に 6 人プレーポーカーでプロプレイヤーに勝っている[4][5].

1.4 セブンティーンポーカー

本研究で扱うセブンティーンポーカーは漫画「LAIR GAME」8 巻に出てくるカードゲームである．LAIR GAME はドラマとしても放映されているが，劇中のゲームであるため、セブンティーンポーカーはあまり認知されておらず分析もされていない．一般的なポーカーは我慢のゲームと言われている．役が揃えにくく，強い役が出るまでひたすら勝負を降りることが必要だからだ．しかしこのセブンティーンポーカーは使用する枚数が 17 枚と少ないので誰でも役が揃えやすく，短時間でスリルのある駆け引きを楽しむことができる．そこで，このゲームのルールと勝ち方を調べると共に少しでも認知され遊んでいただきたいと思い，この研究をすることにした．

1.5 本研究の目的

セブンティーンポーカーはマイナーなゲームであるため，強い AI は存在しない．そこで本研究ではセブンティーンポーカーにおける最適な手札の交換，チップの賭け方，ゲームを降りるタイミングを模索し，勝率が高い AI を目指す．

1.6 本報告書の構成

2 節では本研究で扱うセブンティーンポーカーのルール，作成した AI とその戦略，そして実験を行った結果と考察を述べる．3 節では結論と実験結果から見たセブンティーンポーカーAI の今後の課題を述べる．

2. 研究内容

2.1 セブンティーンポーカーのルール

本研究で扱うセブンティーンポーカーのルールを説明する．セブンティーンポーカーのルールを以下に列挙する．

- 対戦人数は二人
- 初期のチップ数は 100 枚
- カードは各スートの A，J，Q，K とジョーカーの計 17 枚を使用
- 数字の強さは J<Q<K<A で，ジョーカーは任意のカードに変えられる．
- カードはゲームを行う前にシャッフルしておく．
- 役は弱い順からワンペア，ツーペア，スリーカード，ストレート，フルハウス，フォーカード，ロイヤルストレートフラッシュ，ファイブカード

まずプレイヤーはアンティ(場代)としてチップ 5 枚を場に出し，手札を 5 枚貰う．
次に一人ずつ 1st ベットを行う．1st ベットでは次の中から一つを選択する．

- 「チェック」
  賭けをパスする．自分の役が弱い場合や，相手の出方を見たい場合に選択する．
- 「ベット」
  賭け金を設定する．最低ベット数は 5 枚で，上限は 15 枚である．

両者が「チェック」をした場合は勝負不成立となり，アンティはディーラーに回収される．どちらかが「ベット」をした場合は次の中から一つを交互に選択する．

- 「レイズ」
  賭け金をつり上げる．上限は 15 枚とする．
- 「コール」
  賭け金を揃える．
- 「フォルド」
  勝負を降りる．場にあるアンティはディーラーに回収され，この行動をするまでに賭けられたチップは相手のものになる．
- 「オールイン」
  所持しているチップを全て賭ける．コールするためのチップは足りないが勝負したい場合に用いる．賭けるチップ数を合わせるために，相手が賭けていたチップとオールインしたチップの差額は返却される．

これらはどちらかが「コール」「オールイン」「フォルド」を選択するまで行う．もし「コール」「オールイン」を選択した場合勝負成立となり，手札交換に移る．

手札交換は任意の枚数を交換できる．交換したカードは山札の下に入れられる．手札交換を行なった後，2nd ベットに移る．1st ベットで「オールイン」を選択していた場合 2nd ベットは行わない．

2nd ベットは 1st ベットと同様のことを行うが，最低ベット数は 1st ベットで最終的に設定された枚数で，上限は 30 枚とする．

最後にお互いの手札を公開して役が強い方がアンティを含む場にあるチップを獲得できる．役が同じ場合は揃っている役の数字の強さを比較し，勝敗を決める．数字の強さも同じ場合は引き分けとなり，アンティと賭けたチップは戻される．これを 10 ゲーム行い，チップ数の多い方が勝利となる．

原作ではゲームを行う前のカードは一般的なヒンズーシャッフルと，カードの山を半分ずつ持ち交互にかみ合わせて混ぜるリフルシャッフルを 1 回ずつ行う．その後両者が 1 回ずつ「カット」を行う．「カット」は山札の上から何枚目かを指定し，指定された場所で山札を切り分けて重ねる．ゲーム中では任意のタイミングで山札をリフルシャッフルできる．これらのシャッフルは規則正しく行われ，山札を操作して引きたいカードを引くことができてしまうため本研究では原作と異なる上記のルールで行う．

2.2 セブンティーンポーカーAI

本研究では，Java 言語を用いてセブンティーンポーカーAI を作成する．

図 1 に本研究で作成した AI の実行の様子を示す．

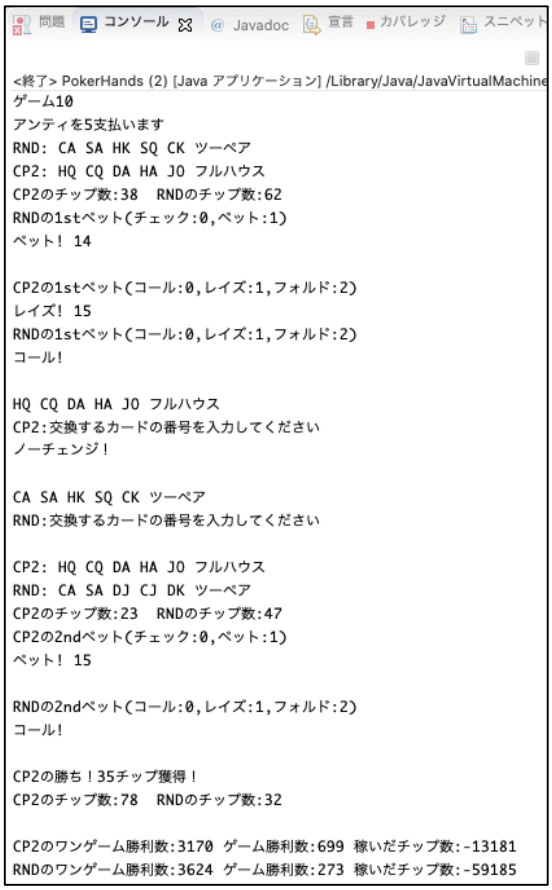

図 1　AI 実行の様子

AI は 3 種類あり，名前は「CP1」「CP2」「RND」とした．AI の戦略は次の節で説明する．これらの中から二つ選び，名前を引数とするプレイヤーのコンストラクタを作成し，実行する．そうするとまずゲーム数と支払うアンティが出力される．次に各プレイヤーの情報とゲーム中の 1st ベットの行

動選択，手札交換，2nd ベットの行動選択の様子が出力される．次に 1 ゲームの対戦結果が出力される．これらが 1000 回ループされ，最後に各プレイヤーの戦績が出力される．

2.3 セブンティーンポーカーAI の戦略

2.1 で述べたルールにおける役が完成する確率，役の強さ，相手の賭け金を評価値として作成した評価基準が異なる 3 種類の AI「CP1」「CP2」「RND」の戦略について説明する．

全ての AI の手札交換は同じ方法を用いる．まず表 3 にセブンティーンポーカーの役と各役が成立する確率を示す．

表 3　セブンティーンポーカーの役と成立する確率

役	役の説明	確率
ファイブカード	同じ数字のカードが 5 枚揃う	0.04%
ロイヤルストレートフラッシュ	同じスートかつ数字が連続して揃う	0.07%
フォーカード	同じ数字のカードが 4 枚揃う	3.9%
フルハウス	同じ数字のカードが 3 枚と同じ数字のカードが 2 枚揃う	8.21%
ストレート	数字が連続して揃う	4.24%
スリーカード	同じ数字のカードが 3 枚揃う	31.11%
ツーペア	同じ数字のカード 2 枚組が 2 組揃う	27.96%
ワンペア	同じ数字のカードが 2 枚揃う	24.47%

表 3 に示されるように，セブンティーンポーカーでも Wild Card Poker Paradox，すなわち役の強弱と役の成立確率が入れ替わる現象が発生する．セブンティーンポーカーでは，スリーカード，ツーペア，ワンペアが成立確率の大部分を占めているが，この三つの役の中では強い役の順に確率が高い．これはツーペアとワンペアはジョーカーがあると揃わない役になっているためである．この役が揃ったとき相手がジョーカーを持っている可能性が高い．ストレートはジョーカー必須のため，より強い役であるフルハウスよりも確率は低い．ファイブカードとロイヤルストレートフラッシュは成立する確率は低いが，ジョーカーが必須なので揃えることができれば相手はこれらを揃えることができないので勝ちが確定する役となっている．これらのことから一枚しかないジョーカーを引けるかどうかがかなり重要となっている．以上を踏まえて考えた手札交換の方法を表 4 に示す．

表 4　各役の交換方法

役	交換方法
ストレート以上の役	交換しない
スリーカード	揃っているトリオを残してその他を交換する
ツーペア	揃っているペアの強い方を残してその他を交換する
ワンペア	揃っているペアを残してその他を交換する

ファイブカード，ロイヤルストレートフラッシュは勝ちが確定しているので交換しない．フルハウス，ストレートは完成しにくく，カードを交換すると弱い役ができてしまう可能性があるため交換しない．スリーカードはフルハウス，フォーカード，ファイブカードを狙って交換する．ツーペアはペア二組を残して一枚交換するよりも，強いペア一組を残して 3 枚交換した方がより強い役を揃えやすく，ジョーカーを引きやすいため表のような交換方法となっている．ワンペアはツーペア，スリーカード，フルハウス，フォーカード，ファイブカードを狙って交換する．

次に各 AI の行動選択方法について説明する．まず「CP1」の行動選択方法を表 5 に示す．「CP1」は基本的に勝負し，強い役が出れば出るほど賭け金を吊り上げてより多く稼ぐという特徴を持つ AI である．

表 5 「CP1」の行動選択

ラウンド	役	条件	行動
1st ベット (チェック or ベット)	全て	所持チップ数＝最低ベット数	5 チップベット
	スリーカード以下	なし	5 チップベット
	ストレート以上	なし	5～6 チップベット
1st ベット (どちらかがベット後)	全て	コールするためのチップが足りない	オールイン
	スリーカード以下	最低ベット数 8 未満	コール
		最低ベット数 8 以上	フォルド
	ストレート以上	最低ベット数 10 または 1～2 チップレイズ不可能	コール
		1～2 チップレイズ可能	1～2 チップレイズ (最大 2 回)
2nd ベット (チェック or ベット)	ツーペア以下		チェック
	スリーカード	なし	最低ベット数ベット
	スリーカード以上	所持チップ数<最低ベット数+3	最低ベット数ベット
	ストレート以上	なし	最低ベット数+1～3 ベット
2nd ベット (どちらかがベット後)	全て	コールするためのチップが足りない	オールイン
	ツーペア以下	なし	フォルド
	スリーカードまたはストレート	最低ベット数 15 未満	コール
		最低ベット数 15 以上	フォルド
	フルハウスまたはフォーカード	最低ベット数 15 または 1 チップレイズ不可能	コール
		1 チップレイズ可能	1 チップレイズ (最大 2 回)
	ロイヤルストレートフラッシュ以上	最低ベット数 30 または 2 チップレイズ不可能	コール
		2 チップレイズ可能	2 チップレイズ (最大 2 回)

1st ベットのチェックかベットを選択する場面ではどんな役でもベットする．手札交換でより強い役ができる可能性が高いため，必ず勝負をするようにしている．ベット後の選択は主に 2 つの条件から考える．スリーカード以下は成立する確率が高いので基本的にコールし，ストレート以上はレイズする．続いて 2nd ベットのチェックかベットを選択する場面ではツーペア以下はチェックをし，それ以外はベットする．手札交換後のツーペア以下はかなり弱い役なのでこの時点で賭けることをやめ，失うチップを少なくするためである．ベット後の選択は主に 4 つの条件から考える．ツーペア以下は先に述べた理由からフォルドする．スリーカード，ストレートはコールする．フルハウス，フォーカードはレイズする．ロイヤルストレートフラッシュ以上は勝ちが確定しているのでより大きくレイズする．

次に「CP2」の行動選択方法を表 5 に示す．「CP2」は「CP1」よりも評価基準がシンプルかつ賭け金が控えめで，失うチップをできるだけ抑えるという特徴を持つ AI である．

表 6 「CP2」の行動選択

ラウンド	役	条件	行動
1st ベット (チェック or ベット)	ツーペア以下	なし	チェック
	ストレート以上	なし	5 チップベット
1st ベット (どちらかがベット後)	全て	コールするためのチップが足りない	オールイン
	スリーカード以下	最低ベット数 8 未満	コール
		最低ベット数 8 以上	フォルド
	ストレート以上	最低ベット数 10 または 1 チップレイズ不可能	コール
		1 チップレイズ可能	1 チップレイズ(最大 2 回)
2nd ベット (チェック or ベット)	ツーペア以下	なし	チェック
	ストレート以上	なし	5 チップベット
2nd ベット (どちらかがベット後)	全て	コールするためのチップが足りない	オールイン
	ツーペア以下	なし	フォルド
	スリーカード	最低ベット数 15 未満	コール
		最低ベット数 15 以上	フォルド
	ストレート以上	最低ベット数 15 または 1 チップレイズ不可能	コール
		1 チップレイズ可能	1 チップレイズ(最大 2 回)

「CP1」と異なる点は主に 3 つある．1st ベットと 2nd ベットのチェックかベットを選択する場面ではツーペア以下はチェックする．1st ベットのベット後ストレート以上の場合は 1 チップずつレイズする．2nd ベットのベット後の賭けるチップの上限が低く，レイズする場合も 2 チップ以上のレイズをしない．これらのことから「CP2」よりも勝負をせず，失うチップが少ない戦略となっている．

3 つ目の AI「RND」の行動選択は選択するためのチップ数が足りない場合を除いて，ランダムとなっている．弱い役でも賭け金を大きく吊り上げることがあるためブラフが発生する．

2.4 セブンティーンポーカーAI プログラム

付録に本研究で作成したセブンティーンポーカーAI のソースプログラムを示す．

Pokerhands はセブンティーンポーカーを行うクラスである．各メソッドの説明を次に示す．

- getPokerHand(cards:List<Integer>型):int 型
  プレイヤーのカード 5 枚のリストから役を判定する
- getStringNumber(num:int 型):String 型
  数字を文字列に変換する
- showName(player:Player 型):void 型
  プレイヤーの名前を表示する
- showCards(cards5:List<Integer>型):int 型
  カード 5 枚と役を表示する
- change(player:Player 型):List<Integer>型
  プレイヤーの手札を自動的に交換する
- finalJudge(player1: Player 型, player2:Player 型):int 型
  1 ゲームの勝敗判定を行う

- showChip(player1:Player 型, player2:Player 型):void 型
  両プレイヤーの所持チップ数を表示する

CardManager はトランプカードを操作するクラスである．各メソッドの説明を次に示す．

- organize():void 型
  山札のリストを初期化し，17 枚のカードを格納する
- shuffle():void 型
  山札をシャッフルする
- getCard(num:int 型):int 型
  山札からカードを引く
- returnCard(cards5:List<Integer>型, changeCard:List<Integer>型):void 型
  交換するカードを山札に戻す

Player はプレイヤーを表すクラスである．各メソッドの説明を次に示す．

- setName(name:String 型):void 型
  プレイヤー名を設定する
- getName():String 型
  プレイヤー名を取得する
- getChip():int 型
  所持チップ数を取得する
- betAnte():int 型
  アンティを支払う
- getCards5():List<Integer>型
  手札を保存する
- setCards5(cards:List<Integer>型):void 型
  手札を取得する
- updateCards5(cards:List<Integer>型):void 型
  手札を更新する
- getHand():int 型
  役を取得する
- setHand(hand:int 型):void 型
  役を保存する
- bet(bet_value:int 型):int 型
  与えられたチップ数を賭ける
- allBet(bet_value:int 型):void 型
  所持チップ数を全て賭ける
- diffelence(min:int 型):boolean 型
  所持ベット数が最低ベット数に足りているかを判定する
- getTotal():int 型
  合計ベット数を取得する
- back(value:int 型):void 型
  与えられた数のチップを払い戻す
- earn(value:int 型):void 型
  チップを獲得する
- lostC():void 型
  失ったチップをカウントする
- anteLostC():void 型
  アンティを失ったチップ数としてカウントする

- raiseT()：boorean 型
  レイズが 2 回目かどうかを判定する
- raiseC()：void 型
  レイズの回数をカウントする
- winC()：void 型
  1 ゲームの勝利数をカウントする
- resetTotal()：void 型
  賭け金の合計とレイズの回数をリセットする
- resetCards()：void 型
  手札をリセットする
- resetChip()：void 型
  所持チップ数をリセットする
- resetLost()：void 型
  失ったチップ数をリセットする
- oneWinRate()：int 型
  1 ゲーム勝利数を取得する
- winRate()：int 型
  ゲーム勝利数を取得する
- getEarn()：int 型
  稼いだチップ(得たチップ数-失ったチップ数)を取得する

2.5 実験結果・考察

2.3で述べたAI同士を1000回させた結果を表7に示す.CP1とCP2の対戦では勝率に大きな差は出なかったが，CP1とRND，CP2とRND の対戦ではどちらもRNDの勝率が低いという結果になった.まずCP1の勝率が高かった要因は，RNDのブラフをある程度対処できていたことだと考えられる．次にCP2の勝率が高かった要因はRNDの極端なブラフを対処できなかったが，安定してチップを獲得していたことが考えられる．RNDに対するCP1とCP2 の勝率に差ができたのは，CP2とRNDの対戦で稼いだチップ数の差がCP1とRNDの対戦で稼いだチップ数より大きいことから，賭け金の大きさが関係していることが分かった.

表 7　各対戦の勝率と稼いだチップ(得た数ー失った数)

	CP1	CP2	RND
CP1		50%/-9115	62%/-17913
CP2	50%/-10468		71%/-12359
RND	38%/-49007	29%/-58792	

以下ではそれぞれの結果について統計的に検証する.

2 人プレイであるので，各 AI の戦略に優劣が無ければ，勝率は 50%である．そこで，勝率が 50%と仮定した時に，統計上有意な差があるかを検証する．勝率 p の勝負を N 回行った場合，標準偏差 s は以下の式で表される．

$$s = \sqrt{N * p * (1 - p)}$$

p=0.5 と仮定すると，N=1000 ならば標準偏差は

$$\sqrt{1000 * 0.5 * 0.5} = 15.8$$

となる．信頼区間 95%となるのは勝利回数が平均値からの差が 15.8*19.6=31.0 となる区間である．

したがって，勝率が 50%ならば，1000 試合すれば 95%の確率で勝利回数は 500±31.0，勝率は 50±3.1%に収まる．

以上を踏まえて改めて表 7 の結果を見ると CP1 と CP2 が対戦したときの勝率は 50±3.1%に収まっており，統計上有意な差は無かった．よって CP1 と CP2 の戦略に優劣はないことが分かる．次に CP1 と RND，CP2 と RND が対戦対戦したときの勝率は 50±3.1%に収まっておらず，統計上有意な差があった．

先に述べたように CP1 と CP2 の戦略に優劣は無かったため，RND が他の戦略よりはるかに劣っていることが分かる．

3. 結論・今後の課題

本研究では，Java 言語を用いてセブンティーンポーカーにおける AI を開発した．基本的に勝負し，強い役が出れば出るほど賭け金を吊り上げてより多く稼ぐ CP1 と，CP1 よりも評価基準がシンプルかつ賭け金が控えめで，失うチップをできるだけ抑える CP2 の 2 つの AI を作成したが，CP1 も CP2 も全ての対戦で高い勝率が出なかったためどちらも強い AI とは言えなかった．今後より強い AI にするための課題としては，相手のブラフを対処しつつ使いこなす戦略，大きな負けをしない戦略を立てることである．また今回行わなかった人間が対戦するときの行動選択を分析することが挙げられる．

謝辞

本論文の執筆にあたり，多くの方々にご支援いただきました．

中間審査では，溝渕昭二准教授より，貴重なご指導とご助言を賜りました．感謝申し上げます．

主指導教員である石水隆講師には，研究の着想から，調査，論文執筆まで多くのご指導をいただきました。心から感謝申し上げます．

最後に，所属する情報論理工学研究室のみなさまには多くのご支援をいただきました．お礼申し上げます．

ありがとうございました．

参考文献

[1] 甲斐谷忍, LAIR GAME 8, 集英社 (2012)

[2] Byron Spice, Carnegie Mellon Artificial Intelligence Beats Top Poker Pros, Carnegie Mellon University, (2017/7/31)
https://www.cmu.edu/news/stories/archives/2017/january/AI-beats-poker-pros.html

[3] 木原直哉, なぜ人間はポーカーでAIに負けたのか？　日本トッププロが解説する“違和感”, IT media (2017/2/3) https://www.itmedia.co.jp/pcuser/articles/1702/03/news028.html

[4] Noam Brown ans Tuomas Sandholm, Superhuman AI for multiplayer poker, Computer Science, No.365, pp.885-890 (2019) https://science.sciencemag.org/content/365/6456/885

[5] Tom Simonite, ポーカーの複数人対戦でAIがプロに圧勝、初の快挙が世界にもたらすインパクト, Business, Wired (2019/7/29)
https://wired.jp/2019/07/29/new-poker-bot-beat-multiple-pros/

[6] Stave Gadbois, ″Poker with wild cards - a paradox?″, Mathenatics Magazine, No,69, pp. 283-285 (1996) https://www.maa.org/sites/default/files/Steve_Gadbois22042.pdf

付録

- Pokerhands クラス

```
package cal;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.Scanner;

/**
 * セブンティーンポーカーを行うクラス
 *
 */
public class PokerHands {
  // エラー
  static final int ERROR          = 0;

  // ポーカーの役
  // ファイブカード
    static final int  FIVE_OF_A_KIND    = 1;
  // ロイヤルフラッシュ（ロイヤルストレートフラッシュ）
  static final int ROYAL_FLUSH     = 2;
  // フォーカード
  static final int FOUR_OF_A_KIND  = 3;
  // フルハウス
  static final int FULL_HOUSE      = 4;
  // ストレート
  static final int STRAIGHT        = 5;
  // スリーカード
  static final int THREE_OF_A_KIND = 6;
  // ツーペア
  static final int TWO_PAIR        = 7;
  // ワンペア
  static final int ONE_PAIR        = 8;



  // カード５枚を配列で渡して役を判定するメソッド
  static int getPockerHand( List<Integer> cards )
  {
    // 配列が null だったらエラー
    if ( null == cards ) return ERROR;

    // 配列の個数が 5 でなければエラー
    if ( 5 != cards.size() ) return ERROR;
```

```
// トランプのマークの個数を格納する配列
int[] suit = new int[ 5 ];

// トランプのマークの個数に 0 を代入（初期化）
for ( int i = 0; i < suit.length; i++ )
  suit[ i ] = 0;


// トランプの番号(1-5)の個数を格納する配列
int[] number = new int[ 5 ];

// トランプのマークの個数に 0 を代入（初期化）
for ( int i = 0; i < number.length; i++ )
  number[ i ] = 0;

// ５枚のカードのマークと番号の個数を格納
for ( int i = 0; i < cards.size(); i++ ) {
  // マーク
  int mark = cards.get(i) / 100;
  // 番号
  int num = cards.get(i) % 100;

  // mark が 1 から 4 の範囲外であればエラー
  if ( ( 1 > mark ) || ( 5 < mark ) ) {
    return ERROR;
  }

  // num が 1 から 13 の範囲外であればエラー
  if ( ( 0 > num ) || ( 5 < num ) ) {
    return ERROR;
  }

  if(mark == 5)
    ++ suit[4];
  else
  // マークの個数に１を足す
  ++ suit[ mark - 1 ];

  if(num == 0)
    ++ number[4];
  else
    // 番号の個数に１を足す
    ++ number[ num - 1 ];
}

// 番号と番号の個数の最大値を取得
```

```
int number_max = 0;
for ( int i = 0; i < number.length - 1; i++ ) {
  if ( number_max < number[ i ] )
    number_max = number[ i ];
}

boolean joker_check = false;
// もしジョーカーがあるかどうか
if(number[4] == 1)
  joker_check = true;

// ここから判定処理

// 個数の最大が 4 の場合、フォーカード or ファイブカード
if ( 4 == number_max )
  if(joker_check)
    return FIVE_OF_A_KIND;
  else
    return FOUR_OF_A_KIND;



// マークの個数の最大値を取得
// ５枚のカードが同じマークの場合、suit_max=5 となる
int suit_max = 0;
for ( int i = 0; i < suit.length; i++ ) {
  if ( suit_max < suit[ i ] )
    suit_max = suit[ i ];
}

// ストレートの判定
boolean isStraight = false;
int continuous1 = 0;
for ( int i = 0; i < number.length; ++ i ) {
  if ( 1 != number[ i ] ) {
    continuous1 = 0;
  }
  else {
    ++ continuous1;

    // ５回連続だったら
    if ( 5 == continuous1 ) {
      // ストレートは確定
      isStraight = true;
      break;
    }
  }
```

```
        }

        // マークが全て同じで、ストレートだったら
        if ( ( 4 == suit_max ) && isStraight ) {
          // ロイヤルストレートフラッシュ確定
          return ROYAL_FLUSH;

        }

        // 個数の最大が３で、もう１つペアが存在したら
        if ( 3 == number_max ) {
          for ( int i = 0; i < number.length - 1; ++ i ) {
            if ( 2 == number[ i ] ) {
              // フルハウス確定
              return FULL_HOUSE;
            }
          }
        }


        // ストレートだったら
        if ( isStraight ) {
          return STRAIGHT;
        }

        // 個数の最大が３の場合、スリーカードかフォーカード
        if ( 3 == number_max )
          if(joker_check)
            return FOUR_OF_A_KIND;
          else
            return THREE_OF_A_KIND;

        // ペアの個数
        int pair_num = 0;
        for ( int i = 0; i < number.length - 1; ++ i ) {
          if ( 2 == number[ i ] )
            ++ pair_num;
        }

        // ペアが２つであれば
        if ( 2 == pair_num )
          if(joker_check)
            return FULL_HOUSE;
          else
            return TWO_PAIR;

        // ペアが１つであれば
```

```
        if ( 1 == pair_num )
          if(joker_check)
            return THREE_OF_A_KIND;
          else
            return ONE_PAIR;

        // ワンペア
        return ONE_PAIR;
      }


      // 番号を文字に変換
      static String getStringNumber( int num ) {
        switch ( num ) {
          case 1:
            return "J";
          case 2:
            return "Q";
          case 3:
            return "K";
          case 4:
            return "A";
          case 0:
            return "O";
        }
        return "";
      }

      //プレイヤーを表示
      public static void showName(Player player) {
        System.out.print(player.getName() + ": ");
      }
      // カードと役を表示
      public static int showCards(List<Integer> cards5) {
        for ( int i = 0; i < 5; ++ i ) {
          // マーク
          int mark = cards5.get(i)  / 100;
          // 番号
          int num = cards5.get(i) % 100;

          // 番号を文字列に変更
          String strnum = getStringNumber( num );

          //  表示
          switch ( mark ) {
            case 1:
              // スペード
```

```
            System.out.print( "S" + strnum );
            break;
          case 2:
            // ハート
            System.out.print( "H" + strnum );
            break;
          case 3:
            // ダイヤ
            System.out.print( "D" + strnum );
            break;
          case 4:
            // クラブ
            System.out.print( "C" + strnum );
            break;
          case 5:
            //ジョーカー
            System.out.print( "J" + strnum );
            break;
        }
        System.out.print( " " );
      }

      // 役を判定
      int hand = getPockerHand( cards5 );
      switch ( hand ) {
        case FIVE_OF_A_KIND:
          System.out.println( "ファイブカード" );
          break;
        case ROYAL_FLUSH:
          System.out.println( "ロイヤルストレートフラッシュ" );
          break;

        case FOUR_OF_A_KIND:
          System.out.println( "フォーカード" );
          break;

        case FULL_HOUSE:
          System.out.println( "フルハウス" );
          break;

        case STRAIGHT:
          System.out.println( "ストレート" );
          break;

        case THREE_OF_A_KIND:
          System.out.println( "スリーカード" );
          break;
```

```
      case TWO_PAIR:
        System.out.println( "ツーペア" );
        break;

      case ONE_PAIR:
        System.out.println( "ワンペア" );
        break;

      default:
        System.out.println( "ノーペア" );
        break;
    }
    return hand;
  }

  //CP の手札交換
  public static List<Integer> change(Player player) {
    int[] number = new int[5];
    // 交換するカードの番号
    List<Integer> changeNum = new ArrayList<Integer>(5);
    // 交換するカード
    List<Integer> changeCard = new ArrayList<Integer>(5);

    // トランプの番号の個数に 0 を代入（初期化）
    for ( int i = 0; i < 5; i++ ) {
      number[ i ] = 0;

    }

    // 5 枚のカードの番号の個数を格納
    for ( int i = 0; i < 5; i++ ) {
      // 番号
      int num = player.getCards5().get(i) % 100;

      if(num == 0)
        ++ number[4];
      else
        // 番号の個数に 1 を足す
        ++ number[ num - 1 ];
    }

    if(player.getHand() != 7) {
      //一枚しかないカードの番号を保存
      for (int i = 0; i < 4; i++) {
        if(number[i] == 1) {
          // 手札
```

```
            changeNum.add(i + 1);
          }
        }

        for ( int i = 0; i < 5; i++ ) {
          for(int j = 0; j < changeNum.size(); j++) {
            if(player.getCards5().get(i) % 100 == changeNum.get(j)) {
              changeCard.add(i + 1);
            }
          }
        }
      } else {
        //強い方のペアの番号を保存
        int s = 0;
        for (int i = 0; i < 4; i++) {
          if(number[i] == 2) {
            s = i + 1;
          }
        }
        changeNum.add(s);

        for ( int i = 0; i < 5; i++ ) {
           if(player.getCards5().get(i) % 100 != changeNum.get(0)) {
             changeCard.add(i + 1);
           }

        }
      }

      return changeCard;


    }

    //勝敗判定
    public static int finalJudge(Player player1, Player player2) {
      if(player1.getHand() < player2.getHand()) {
        return 1;
      } else if(player1.getHand() > player2.getHand()) {
        return 2;
      } else {
        // トランプの番号(1-5)の個数と強さを格納する配列
        int[] number1 = new int[ 6 ];
        int[] number2 = new int[ 6 ];

        // トランプの番号の個数に 0 を代入（初期化）
        for ( int i = 0; i < 5; i++ ) {
```

```
        number1[ i ] = 0;
        number2[ i ] = 0;
      }

      // ５枚のカードの番号の個数を格納
      for ( int i = 0; i < 5; i++ ) {
        // 番号
        int num1 = player1.getCards5().get(i) % 100;
        int num2 = player2.getCards5().get(i) % 100;

        if(num1 == 0)
          ++ number1[4];
        else
          // 番号の個数に１を足す
          ++ number1[ num1 - 1 ];

        if(num2 == 0)
          ++ number2[4];
        else
          // 番号の個数に１を足す
          ++ number2[ num2 - 1 ];
      }

      if(number1[4] == 1) {
        // 一番強い数字のペアを取得
        int number_max = 2;
        int s = 0;
        for ( int i = 0; i < 4; i++ ) {
          if ( number_max <= number1[ i ] )
            number_max = number1[ i ];
              s = i;
        }
        ++number1[s];
      }

      if(number2[4] == 1) {
        // 一番強い数字のペアを取得
        int number_max = 2;
        int s = 0;
        for ( int i = 0; i < 4; i++ ) {
          if ( number_max <= number2[ i ] )
            number_max = number2[ i ];
              s = i;
        }
        ++number2[s];
      }
```

```
            //フォーカード同士の場合
            if(player1.getHand() == 3) {
              for ( int i = 0; i < 4; i++ ) {
                if(number1[i] == 4) {
                  number1[5] += i;
                }
                  if(number2[i] == 4) {
                  number2[5] += i;
                  }
              }
              if(number1[5] == number2[5]) {
                for ( int i = 0; i < 4; i++ ) {
                  if(number1[i] == 1) {
                    number1[5] += i;
                  }
                    if(number2[i] == 1) {
                    number2[5] += i;
                    }
                }
              }
            //フルハウスまたはスリーカード同士の場合
            } else if (player1.getHand() == 4 || player1.getHand() == 6) {
              for ( int i = 0; i < 4; i++ ) {
                if(number1[i] == 3) {
                  number1[5] += i;
                }
                  if(number2[i] == 3) {
                  number2[5] += i;
                  }
              }
            } else {
              for ( int i = 0; i < 4; i++ ) {
                if(number1[i] == 2) {
                  number1[5] += i;
                }
                  if(number2[i] == 2) {
                  number2[5] += i;
                  }
              }
              if(number1[5] == number2[5]) {
                for ( int i = 0; i < 4; i++ ) {
                  if(number1[i] == 1) {
                    number1[5] += i;
                  }
                    if(number2[i] == 1) {
                    number2[5] += i;
                    }
```

```
            }
          }
        }

        //判定
        if(number1[5] > number2[5]) {
          return 1;
        } else if(number1[5] < number2[5]) {
          return 2;
        } else {
          return 0;
        }
      }


    }

    //両プレイヤーのチップ数表示
    private static void showChip(Player player1, Player player2) {
      System.out.println(player1.getName() + "のチップ数:" + player1.getChip() + "  "
  + player2.getName() + "のチップ数:" + player2.getChip());
    }




    // メイン
    public static void main( String[] args ) {
      //スキャナを作成
      Scanner sc = new Scanner(System.in);

      //ランダム
      Random rand = new Random();

      // トランプクラスを作成
      CardsManager cards = new CardsManager();

      // 手札
      List<Integer> cards5 = new ArrayList<Integer>(5);
      //交換するカード
      List<Integer> changeCard = new ArrayList<Integer>(5);

      //プレイヤーを作成
      Player player1 = new Player("CP2");
      Player player2 = new Player("RND");

      Player player = player1;
```

```
int loop = 1000;

for(int k = 0; k < loop; k++) {
  //ゲーム終了判定
  boolean gameSet = false;

  //ゲーム数のカウンタ
  int gameCount = 1;

  while(gameSet == false) {

    //ワンゲーム終了判定
    boolean oneGame = false;
    //アンティ
    int ante = 0;

    //賭けたチップの合計
    int total_chip = 0;

    //最低ベット数
    int min_bet = 5;

    //不確定のベット
    int uncertain = 0;

    //1st ベットのチェック数(2 になると勝負不成立)
    int check1st = 0;

    //2nd ベットのチェック数(2 になると勝負不成立)
    int check2nd = 0;

    //1st ベット判定
    boolean bet1st = false;

    //2nd ベット判定
    boolean bet2nd = false;

    //2st ベットを行うかどうか
    boolean move2nd = true;

    //1st ベットでコール判定
    boolean call1st = false;

    //2nd ベットでコール判定
      boolean call2nd = false;
```

```
  //ゲーム数の表示
System.out.println("ゲーム" + gameCount);

//ゲーム数の表示
System.out.println("アンティを5支払います");


// トランプをシャッフル
cards.shuffle();

//プレイヤーの順番を設定
if(gameCount % 2 == 0) {
  player = player2;
} else {
  player = player1;
}


//0ターン目(ゲームの準備)
for ( int j = 0; j < 2; ++ j ) {
  //アンティを支払う
  ante += player.betAnte();

  // トランプを上から順番に引いていく
  int card_num = 0;
  for ( int i = 0; i <= 17; ++ i ) {
    // 一番上のカードを取得
    int card = cards.getCard( 1 );

    // カードが5枚になったらループを抜ける
    cards5.add(card);
    ++ card_num;
    if ( 5 == card_num ) break;
  }

  //手札を保存
  player.setCards5(cards5);

  //プレイヤーを表示
  showName(player);

  // カードと役を表示
  player.setHand(showCards(player.getCards5()));

  //プレイヤー交代
  if(player == player1) {
    player = player2;
```

```
    } else {
      player = player1;
    }

    cards5.clear();
  }

  //両者のチップ数を表示
  showChip(player1,player2);



  //1st ベット
  while(true) {
    System.out.println(player.getName() + "の 1st ベット(チェック:0,ベット:1)");
    if(player.getName().equals("CP1")) {
      if(player.getHand() > 5 || player.getChip() == 5) {
        System.out.println("ベット! 5");
        player.bet(5);
        min_bet = 5;
        uncertain = 5;
        total_chip += 5;
        bet1st = true;
        System.out.println("");
      } else {
        int bet_value = rand.nextInt(1) + 5;
        System.out.println("ベット!" + bet_value);
        player.bet(bet_value);
        min_bet = bet_value;
        uncertain = bet_value;
        total_chip += bet_value;
        bet1st = true;
        System.out.println("");
      }

    } else if(player.getName().equals("CP2")) {
      if(player.getHand() > 6) {
        System.out.println("チェック!");
        System.out.println("");
        ++check1st;
      } else {
        System.out.println("ベット! 5");
        player.bet(5);
        min_bet = 5;
        uncertain = 5;
        total_chip += 5;
        bet1st = true;
```

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

    } else if(player.getName().equals("RND")) {
      int act = rand.nextInt(2);
      if(act == 0 || min_bet >= player.getChip()) {
        System.out.println("チェック!");
        System.out.println("");
        ++check1st;
      } else {
        int bet_value;
        if(player.getChip() > 15) {
          bet_value = rand.nextInt(10) + 5;
        } else {
          bet_value = rand.nextInt(player.getChip() - 5) + 5;
        }
        System.out.println("ベット! " + bet_value);
        player.bet(bet_value);
        min_bet = bet_value;
        uncertain = bet_value;
        total_chip += bet_value;
        bet1st = true;
        System.out.println("");
      }
    } else {
      while(true) {
        int act = sc.nextInt();
        if(act == 0) {
          System.out.println("チェック!");
          System.out.println("");
          ++check1st;
          break;
        } else if(act == 1) {
          while(true) {
            System.out.println("ベット数を入力してください(5~15)");
            int bet_value = sc.nextInt();
            if(5 > bet_value || 15 < bet_value) {
              System.out.println("エラー");
              System.out.println("");
            } else if(bet_value > player.getChip()){
              System.out.println("チップが足りません");
              System.out.println("");
            } else {
              player.bet(bet_value);
              min_bet = bet_value;
              uncertain = bet_value;
              total_chip += bet_value;
```

```
                bet1st = true;
                System.out.println("");
                break;
              }
            }
            break;
          } else {
            System.out.println("0か1を入力してください");
          }
        }
      }

      //プレイヤー交代
      if(player == player1) {
        player = player2;
      } else {
        player = player1;
      }

      //どちらかがベットした場合
      if(bet1st) {
        break;
      }

      //両方ともチェックした場合
      if(check1st == 2) {
        System.out.println("勝負不成立");
        showChip(player1, player2);
        player1.lostC();
        player2.lostC();
        System.out.println("");
        oneGame = true;
        break;
      }

    }



    if(oneGame == false) {
      while(true) {
        System.out.println(player.getName() + "の1stベット(コール:0,レイズ:1,フォ
        ルド:2)");
        if(player.getName().equals("CP1")) {
          if(!player.diffelence(min_bet)) {
            System.out.println("オールイン!");
            System.out.println("");
```

```
                    total_chip += player.getChip();
                    player.allBet(player.getChip());
                    //差額を返す
                    if(player == player1) {
                      player2.back(min_bet - player.getTotal());
                      total_chip -= (min_bet - player.getTotal());
                      min_bet = player.getTotal();
                    } else {
                      player1.back(min_bet - player.getTotal());
                      total_chip -= (min_bet - player.getTotal());
                      min_bet = player.getTotal();
                    }
                    call1st = true;
                    move2nd = false;
                  } else if(player.getHand() > 5) {
                    if(min_bet < 8) {
                      System.out.println("コール!");
                      System.out.println("");
                      total_chip += player.bet(min_bet);
                      call1st = true;
                    } else {
                      if(player == player1) {
                        System.out.println("フォルド!");
                        System.out.println(player2.getName() + "の勝ち！" + (total_chip -
                        uncertain) + "チップ獲得！");
                        player2.back(uncertain);
                        player2.earn(total_chip - uncertain);
                        player1.lostC();
                        player2.anteLostC();
                        showChip(player1, player2);
                        System.out.println("");
                      } else {
                        System.out.println("フォルド!");
                        System.out.println(player1.getName() + "の勝ち！" + (total_chip -
                        uncertain) + "チップ獲得！");
                        player1.back(uncertain);
                        player1.earn(total_chip - uncertain);
                        player2.lostC();
                        player1.anteLostC();
                        showChip(player1, player2);
                        System.out.println("");
                      }
                      oneGame = true;
                    }
                  } else {
                    if(min_bet == 10 || !player.diffelence(min_bet + 2) ||
                    player.raiseT()) {
```

```
                    System.out.println("コール!");
                    System.out.println("");
                    total_chip += player.bet(min_bet);
                    call1st = true;
                  } else {
                    int bet_value = rand.nextInt(1) + (min_bet + 1);
                    System.out.println("レイズ! " + bet_value);
                    min_bet = bet_value;
                    int bet = player.bet(bet_value);
                    total_chip += bet;
                    uncertain = bet;
                    player.raiseC();
                  }
                }
              } else if(player.getName().equals("CP2")) {
                if(!player.diffelence(min_bet)) {
                  System.out.println("オールイン!");
                  System.out.println("");
                  total_chip += player.getChip();
                  player.allBet(player.getChip());
                  //差額を返す
                  if(player == player1) {
                    player2.back(min_bet - player.getTotal());
                    total_chip -= (min_bet - player.getTotal());
                    min_bet = player.getTotal();
                  } else {
                    player1.back(min_bet - player.getTotal());
                    total_chip -= (min_bet - player.getTotal());
                    min_bet = player.getTotal();
                  }
                  call1st = true;
                  move2nd = false;
                } else if(player.getHand() > 5) {
                  if(min_bet < 8) {
                    System.out.println("コール!");
                    System.out.println("");
                    total_chip += player.bet(min_bet);
                    call1st = true;
                  } else {
                    if(player == player1) {
                      System.out.println("フォルド!");
                      System.out.println(player2.getName() + "の勝ち！" + (total_chip -
                      uncertain) + "チップ獲得！");
                      player2.back(uncertain);
                      player2.earn(total_chip - uncertain);
                      player1.lostC();
                      player2.anteLostC();
```

```
                    showChip(player1, player2);
                    System.out.println("");
                  } else {
                    System.out.println("フォルド!");
                    System.out.println(player1.getName() + "の勝ち！" + (total_chip -
                    uncertain) + "チップ獲得！");
                    player1.back(uncertain);
                    player1.earn(total_chip - uncertain);
                    player2.lostC();
                    player1.anteLostC();
                    showChip(player1, player2);
                    System.out.println("");
                  }
                  oneGame = true;
                 }
                } else {
                 if(min_bet == 10 || !player.diffelence(min_bet + 1) ||
                 player.raiseT()) {
                  System.out.println("コール!");
                  System.out.println("");
                  total_chip += player.bet(min_bet);
                  call1st = true;
                 } else {
                  int bet_value = min_bet + 1;
                  System.out.println("レイズ! " + bet_value);
                  min_bet = bet_value;
                  int bet = player.bet(bet_value);
                  total_chip += bet;
                  uncertain = bet;
                  player.raiseC();
                 }
                }
               } else if(player.getName().equals("RND")) {
                if(!player.diffelence(min_bet)) {
                 System.out.println("オールイン!");
                 System.out.println("");
                 total_chip += player.getChip();
                 player.allBet(player.getChip());
                 //差額を返す
                 if(player == player1) {
                  player2.back(min_bet - player.getTotal());
                  total_chip -= (min_bet - player.getTotal());
                  min_bet = player.getTotal();
                 } else {
                  player1.back(min_bet - player.getTotal());
                  total_chip -= (min_bet - player.getTotal());
                  min_bet = player.getTotal();
```

```
                    }
                    call1st = true;
                    move2nd = false;
                  } else {
                    int act = rand.nextInt(3);
                    if(act == 0 || min_bet > 13 || min_bet + 1 >= player.getChip()) {
                      System.out.println("コール!");
                      System.out.println("");
                      total_chip += player.bet(min_bet);
                      call1st = true;
                    } else if(act == 1) {
                      if(player.getChip() > 15) {
                        int bet_value = rand.nextInt(15 - (min_bet + 1)) + (min_bet +
                        1);
                        System.out.println("レイズ! " + bet_value);
                        min_bet = bet_value;
                        int bet = player.bet(bet_value);
                        total_chip += bet;
                        uncertain = bet;
                      } else {
                        int bet_value = rand.nextInt(player.getChip() - (min_bet + 1))
                        + (min_bet + 1);
                        System.out.println("レイズ! " + bet_value);
                        min_bet = bet_value;
                        int bet = player.bet(bet_value);
                        total_chip += bet;
                        uncertain = bet;
                      }
                    } else {
                      if(player == player1) {
                        System.out.println("フォルド!");
                        System.out.println(player2.getName() + "の勝ち！" + (total_chip -
                        uncertain) + "チップ獲得！");
                        player2.back(uncertain);
                        player2.earn(total_chip - uncertain);
                        player1.lostC();
                        player2.anteLostC();
                        showChip(player1, player2);
                        System.out.println("");
                      } else {
                        System.out.println("フォルド!");
                        System.out.println(player1.getName() + "の勝ち！" + (total_chip -
                        uncertain) + "チップ獲得！");
                        player1.back(uncertain);
                        player1.earn(total_chip - uncertain);
                        player2.lostC();
                        player1.anteLostC();
```

```
                    showChip(player1, player2);
                    System.out.println("");
                  }
                  oneGame = true;
                }
              }
            } else {
              while(true) {
                int act = sc.nextInt();
                if(act == 0) {
                  if(player.diffelence(min_bet)) {
                    System.out.println("コール!");
                    System.out.println("");
                    total_chip += player.bet(min_bet);
                    call1st = true;
                    break;
                  } else {
                    System.out.println("オールイン!");
                    System.out.println("");
                    total_chip += player.getChip();
                    player.allBet(player.getChip());
                    //差額を返す
                    if(player == player1) {
                      player2.back(min_bet - player.getTotal());
                      total_chip -= (min_bet - player.getTotal());
                      min_bet = player.getTotal();
                    } else {
                      player1.back(min_bet - player.getTotal());
                      total_chip -= (min_bet - player.getTotal());
                      min_bet = player.getTotal();
                    }
                    call1st = true;
                    move2nd = false;
                    break;
                  }
                } else if(act == 1) {
                  while(true) {
                    System.out.println("ベット数を入力してください(" + (min_bet+1)
                    +"~15)");
                    int bet_value = sc.nextInt();
                    if((min_bet + 1) > bet_value || 15 < bet_value) {
                      System.out.println("エラー");
                      System.out.println("");
                    } else if(!player.diffelence(bet_value)){
                      System.out.println("チップが足りません");
                      System.out.println("");
                    } else {
```

```
                    min_bet = bet_value;
                    int bet = player.bet(bet_value);
                    total_chip += bet;
                    uncertain = bet;
                    break;
                  }
                }
                break;
              } else if(act == 2){
                if(player == player1) {
                  System.out.println("フォルド!");
                  System.out.println(player2.getName() + "の勝ち！" + (total_chip -
                  uncertain) + "チップ獲得！");
                  player2.back(uncertain);
                  player2.earn(total_chip - uncertain);
                  player1.lostC();
                  player2.anteLostC();
                  showChip(player1, player2);
                  System.out.println("");
                } else {
                  System.out.println("フォルド!");
                  System.out.println(player1.getName() + "の勝ち！" + (total_chip -
                  uncertain) + "チップ獲得！");
                  player1.back(uncertain);
                  player1.earn(total_chip - uncertain);
                  player2.lostC();
                  player1.anteLostC();
                  showChip(player1, player2);
                  System.out.println("");
                }
                oneGame = true;
                break;
              } else {
                System.out.println("0か1か2を入力してください");
              }
            }
          }

          //プレイヤー交代
          if(player == player1) {
            player = player2;
          } else {
            player = player1;
          }

          if(call1st == true) {
            break;
```

```
    }

    if(oneGame == true) {
      break;
    }

  }
}



//カード交換
if(oneGame == false) {
  //プレイヤーの順番を設定
  if(k % 2 == 0) {
    player = player2;
  } else {
    player = player1;
  }
  for ( int j = 0; j < 2; ++ j ) {
    showCards(player.getCards5());
    System.out.println(player.getName() + ":交換するカードの番号を入力してくださ
    い");
    if(player.getName().equals("CP1") || player.getName().equals("CP2") ||
    player.getName().equals("RND")) {
      if(player.getHand() < 6) {
        System.out.println("ノーチェンジ！");
        System.out.println("");
      } else {
        changeCard = change(player);
        //手札を保存
        cards5 = player.getCards5();

        //交換するカードを戻す
        cards.returnCard(cards5, changeCard);


        //カードを取得
        for ( int i = 0; i < changeCard.size(); ++ i ) {
          // 一番上のカードを取得
          int card = cards.getCard( 1 );
          cards5.set(changeCard.get(i) -1, card);
        }

        //手札を更新
        player.updateCards5(cards5);
        System.out.println("");
```

```
            changeCard.clear();
          }
        } else {
          int num = sc.nextInt();
          if(num == 0) {
            System.out.println("ノーチェンジ！");
            System.out.println("");
          } else {
            String[] box = String.valueOf(num).split("");

            for(int i = 0;i < box.length; i++) {
              changeCard.add(Integer.parseInt(box[i]));
            }

            //手札を保存
            cards5 = player.getCards5();

            //交換するカードを戻す
            cards.returnCard(cards5, changeCard);


            //カードを取得
            for ( int i = 0; i < changeCard.size(); ++ i ) {
              // 一番上のカードを取得
              int card = cards.getCard( 1 );
              cards5.set(changeCard.get(i) -1, card);
            }

            //手札を更新
            player.updateCards5(cards5);
            System.out.println("");
            changeCard.clear();
          }
        }

        //プレイヤー交代
        if(player == player1) {
          player = player2;
        } else {
          player = player1;
        }
      }

      //カード交換後
      for ( int j = 0; j < 2; ++ j ) {
        //プレイヤーを表示
        showName(player);
```

```
    // カードと役を表示
    player.setHand(showCards(player.getCards5()));

    //プレイヤー交代
    if(player == player1) {
      player = player2;
    } else {
      player = player1;
    }
  }

  //両者のチップ数を表示
  showChip(player1,player2);
}



//2nd ベット
if(oneGame == false && move2nd == true) {
  player1.resetTotal();
  player2.resetTotal();
  while(true) {
    System.out.println(player.getName() + "の 2nd ベット(チェック:0,ベッ
    ト:1)");
    if(player.getName().equals("CP1")) {
      if(player.getHand() > 6) {
        System.out.println("チェック!");
        System.out.println("");
        ++check2nd;
      } else if (player.getHand() == 6 || player.getChip() < min_bet + 3){
        System.out.println("ベット! " + min_bet);
        player.bet(min_bet);
        total_chip += min_bet;
        uncertain = min_bet;
        bet2nd = true;
        System.out.println("");
      } else {
        int bet_value = rand.nextInt(2) + min_bet + 1;
        System.out.println("ベット!" + bet_value);
        player.bet(bet_value);
        min_bet = bet_value;
        total_chip += bet_value;
        uncertain = bet_value;
        bet2nd = true;
        System.out.println("");
      }
```

```
                }else if(player.getName().equals("CP2")) {
                  if(player.getHand() > 6) {
                    System.out.println("チェック!");
                    System.out.println("");
                    ++check2nd;
                  } else {
                    System.out.println("ベット! " + min_bet);
                    player.bet(min_bet);
                    total_chip += min_bet;
                    uncertain = min_bet;
                    bet2nd = true;
                    System.out.println("");
                  }
                } else if(player.getName().equals("RND")) {
                  int act = rand.nextInt(2);
                  if(act == 0 || min_bet >= player.getChip()) {
                    System.out.println("チェック!");
                    System.out.println("");
                    ++check2nd;
                  } else {
                    int bet_value;
                    if(player.getChip() > 30) {
                      bet_value = rand.nextInt(30 - min_bet) + min_bet;
                    } else {
                      bet_value = rand.nextInt(player.getChip() - min_bet) + min_bet;
                    }
                    System.out.println("ベット! " + bet_value);
                    player.bet(bet_value);
                    min_bet = bet_value;
                    total_chip += bet_value;
                    uncertain = bet_value;
                    bet2nd = true;
                    System.out.println("");
                  }
                } else {
                  while(true) {
                    int act = sc.nextInt();
                    if(act == 0) {
                      System.out.println("チェック!");
                      System.out.println("");
                      ++check2nd;
                      break;
                    } else if(act == 1) {
                      while(true) {
                        System.out.println("ベット数を入力してください(" + min_bet +
                        "~30)");
                        int bet_value = sc.nextInt();
```

```
                    if(min_bet > bet_value || 30 < bet_value) {
                      System.out.println("エラー");
                      System.out.println("");
                    } else if(bet_value > player.getChip()){
                      System.out.println("チップが足りません");
                      System.out.println("");
                    } else {
                      player.bet(bet_value);
                      min_bet = bet_value;
                      total_chip += bet_value;
                      uncertain = bet_value;
                      bet2nd = true;
                      break;
                    }
                  }
                  break;
                } else {
                  System.out.println("0か1を入力してください");
                }
              }
            }

            //プレイヤー交代
            if(player == player1) {
              player = player2;
            } else {
              player = player1;
            }

            //どちらかがベットした場合
            if(bet2nd) {
              break;
            }

            //両方ともチェックした場合
            if(check2nd == 2) {
              System.out.println("勝負不成立");
              showChip(player1, player2);
              player1.lostC();
              player2.lostC();
              System.out.println("");
              oneGame = true;
              break;
            }

          }
        }
```

```
            if(oneGame == false && move2nd == true) {
              while(true) {
                System.out.println(player.getName() + "の 2nd ベット(コール:0,レイズ:1,フォ
                ルド:2)");
                if(player.getName().equals("CP1")) {
                  if(!player.diffelence(min_bet)) {
                    System.out.println("オールイン!");
                    System.out.println("");
                    total_chip += player.getChip();
                    player.allBet(player.getChip());
                    //差額を返す
                    if(player == player1) {
                      player2.back(min_bet - player.getTotal());
                      total_chip -= (min_bet - player.getTotal());
                      min_bet = player.getTotal();
                    } else {
                      player1.back(min_bet - player.getTotal());
                      total_chip -= (min_bet - player.getTotal());
                      min_bet = player.getTotal();
                    }
                    call2nd = true;
                  }else if(player.getHand() > 6) {
                    if(player == player1) {
                      System.out.println("フォルド!");
                      System.out.println(player2.getName() + "の勝ち！" + (total_chip -
                      uncertain) + "チップ獲得！");
                      player2.back(uncertain);
                      player2.earn(total_chip - uncertain);
                      player1.lostC();
                      player2.anteLostC();
                      showChip(player1, player2);
                      System.out.println("");
                    } else {
                      System.out.println("フォルド!");
                      System.out.println(player1.getName() + "の勝ち！" + (total_chip -
                      uncertain) + "チップ獲得！");
                      player1.back(uncertain);
                      player1.earn(total_chip - uncertain);
                      player2.lostC();
                      player1.anteLostC();
                      showChip(player1, player2);
                      System.out.println("");
                    }
                    oneGame = true;
                  }else if(player.getHand() == 5 || player.getHand() == 6) {
                    if(min_bet < 15) {
```

```
                    System.out.println("コール!");
                    System.out.println("");
                    total_chip += player.bet(min_bet);
                    call2nd = true;
                  } else {
                    if(player == player1) {
                      System.out.println("フォルド!");
                      System.out.println(player2.getName() + "の勝ち！" + (total_chip -
                      uncertain) + "チップ獲得！");
                      player2.back(uncertain);
                      player2.earn(total_chip - uncertain);
                      player1.lostC();
                      player2.anteLostC();
                      showChip(player1, player2);
                      System.out.println("");
                    } else {
                      System.out.println("フォルド!");
                      System.out.println(player1.getName() + "の勝ち！" + (total_chip -
                      uncertain) + "チップ獲得！");
                      player1.back(uncertain);
                      player1.earn(total_chip - uncertain);
                      player2.lostC();
                      player1.anteLostC();
                      showChip(player1, player2);
                      System.out.println("");
                    }
                    oneGame = true;
                  }
                } else if(player.getHand() == 3 || player.getHand() == 4) {
                  if(min_bet == 15 || !player.diffelence(min_bet + 1) ||
                  player.raiseT()) {
                    System.out.println("コール!");
                    System.out.println("");
                    total_chip += player.bet(min_bet);
                    call2nd = true;
                  } else {
                    int bet_value = min_bet + 1;
                    System.out.println("レイズ! " + bet_value);
                    min_bet = bet_value;
                    int bet = player.bet(bet_value);
                    total_chip += bet;
                    uncertain = bet;
                    player.raiseC();
                  }
                } else {
                  if(min_bet == 30 || !player.diffelence(min_bet + 2) ||
                  player.raiseT()) {
```

```
                    System.out.println("コール!");
                    System.out.println("");
                    total_chip += player.bet(min_bet);
                    call2nd = true;
                  } else {
                    int bet_value = min_bet + 2;
                    System.out.println("レイズ! " + bet_value);
                    min_bet = bet_value;
                    int bet = player.bet(bet_value);
                    total_chip += bet;
                    uncertain = bet;
                    player.raiseC();
                  }
                 }


               }else if(player.getName().equals("CP2")) {
                 if(!player.diffelence(min_bet)) {
                   System.out.println("オールイン!");
                   System.out.println("");
                   total_chip += player.getChip();
                   player.allBet(player.getChip());
                   //差額を返す
                   if(player == player1) {
                     player2.back(min_bet - player.getTotal());
                     total_chip -= (min_bet - player.getTotal());
                     min_bet = player.getTotal();
                   } else {
                     player1.back(min_bet - player.getTotal());
                     total_chip -= (min_bet - player.getTotal());
                     min_bet = player.getTotal();
                   }
                   call2nd = true;
                 } else if(player.getHand() > 6) {
                   if(player == player1) {
                     System.out.println("フォルド!");
                     System.out.println(player2.getName() + "の勝ち！" + (total_chip -
                     uncertain) + "チップ獲得！");
                     player2.back(uncertain);
                     player2.earn(total_chip - uncertain);
                     player1.lostC();
                     player2.anteLostC();
                     showChip(player1, player2);
                     System.out.println("");
                   } else {
                     System.out.println("フォルド!");
                     System.out.println(player1.getName() + "の勝ち！" + (total_chip -
```

```
                    uncertain) + "チップ獲得！");
                    player1.back(uncertain);
                    player1.earn(total_chip - uncertain);
                    player2.lostC();
                    player1.anteLostC();
                    showChip(player1, player2);
                    System.out.println("");
                  }
                  oneGame = true;
                }else if(player.getHand() == 6) {
                  if(min_bet < 15) {
                    System.out.println("コール!");
                    System.out.println("");
                    total_chip += player.bet(min_bet);
                    call2nd = true;
                  } else {
                    if(player == player1) {
                      System.out.println("フォルド!");
                      System.out.println(player2.getName() + "の勝ち！" + (total_chip -
                      uncertain) + "チップ獲得！");
                      player2.back(uncertain);
                      player2.earn(total_chip - uncertain);
                      player1.lostC();
                      player2.anteLostC();
                      showChip(player1, player2);
                      System.out.println("");
                    } else {
                      System.out.println("フォルド!");
                      System.out.println(player1.getName() + "の勝ち！" + (total_chip -
                      uncertain) + "チップ獲得！");
                      player1.back(uncertain);
                      player1.earn(total_chip - uncertain);
                      player2.lostC();
                      player1.anteLostC();
                      showChip(player1, player2);
                      System.out.println("");
                    }
                    oneGame = true;
                  }
                }  else {
                  if(min_bet == 15 || !player.diffelence(min_bet + 1) ||
                  player.raiseT()) {
                    System.out.println("コール!");
                    System.out.println("");
                    total_chip += player.bet(min_bet);
                    call2nd = true;
                  } else {
```

```
                    int bet_value = min_bet + 1;
                    System.out.println("レイズ! " + bet_value);
                    min_bet = bet_value;
                    int bet = player.bet(bet_value);
                    total_chip += bet;
                    uncertain = bet;
                    player.raiseC();
                  }
                 }

               } else if(player.getName().equals("RND")) {
                 if(!player.diffelence(min_bet)) {
                   System.out.println("オールイン!");
                   System.out.println("");
                   total_chip += player.getChip();
                   player.allBet(player.getChip());
                   //差額を返す
                   if(player == player1) {
                     player2.back(min_bet - player.getTotal());
                     total_chip -= (min_bet - player.getTotal());
                     min_bet = player.getTotal();
                   } else {
                     player1.back(min_bet - player.getTotal());
                     total_chip -= (min_bet - player.getTotal());
                     min_bet = player.getTotal();
                   }
                   call2nd = true;
                 } else {
                   int act = rand.nextInt(3);
                   if(act == 0 || min_bet > 28 || min_bet + 1 >= player.getChip()) {
                     System.out.println("コール!");
                     System.out.println("");
                     total_chip += player.bet(min_bet);
                     call2nd = true;
                   } else if(act == 1) {
                     if(player.getChip() > 30) {
                       int bet_value = rand.nextInt(30 - (min_bet + 1)) + (min_bet +
                       1);
                       System.out.println("レイズ! " + bet_value);
                       min_bet = bet_value;
                       int bet = player.bet(bet_value);
                       total_chip += bet;
                       uncertain = bet;
                     } else {
                       int bet_value = rand.nextInt(player.getChip() - (min_bet + 1))
                       + (min_bet + 1);
                       System.out.println("レイズ! " + bet_value);
```

```
                    min_bet = bet_value;
                    int bet = player.bet(bet_value);
                    total_chip += bet;
                    uncertain = bet;
                  }
                } else {
                  if(player == player1) {
                    System.out.println("フォルド!");
                    System.out.println(player2.getName() + "の勝ち！" + (total_chip -
                    uncertain) + "チップ獲得！");
                    player2.back(uncertain);
                    player2.earn(total_chip - uncertain);
                    player1.lostC();
                    player2.anteLostC();
                    showChip(player1, player2);
                    System.out.println("");
                  } else {
                    System.out.println("フォルド!");
                    System.out.println(player1.getName() + "の勝ち！" + (total_chip -
                    uncertain) + "チップ獲得！");
                    player1.back(uncertain);
                    player1.earn(total_chip - uncertain);
                    player2.lostC();
                    player1.anteLostC();
                    showChip(player1, player2);
                    System.out.println("");
                  }
                  oneGame = true;
                }
              }
            } else {
              while(true) {
                int act = sc.nextInt();
                if(act == 0) {
                  if(player.diffelence(min_bet)) {
                    System.out.println("コール!");
                    System.out.println("");
                    total_chip += player.bet(min_bet);
                    call2nd = true;
                    break;
                  } else {
                    System.out.println("オールイン!");
                    System.out.println("");
                    total_chip += player.getChip();
                    player.allBet(player.getChip());
                    //差額を返す
                    if(player == player1) {
```

```
                player2.back(min_bet - player.getTotal());
                total_chip -= (min_bet - player.getTotal());
                min_bet = player.getTotal();
              } else {
                player1.back(min_bet - player.getTotal());
                total_chip -= (min_bet - player.getTotal());
                min_bet = player.getTotal();
              }
              call2nd = true;
              break;
            }
          } else if(act == 1) {
            while(true) {
              System.out.println("ベット数を入力してください(" + (min_bet + 1)
              +"~30)");
              int bet_value = sc.nextInt();
              if((min_bet + 1) > bet_value || 30 < bet_value) {
                System.out.println("エラー");
                System.out.println("");
              } else if(!player.diffelence(bet_value)){
                System.out.println("チップが足りません");
                System.out.println("");
              } else {
                min_bet = bet_value;
                int bet = player.bet(bet_value);
                total_chip += bet;
                uncertain = bet;
                break;
              }
            }
            break;
          } else if(act == 2){
            if(player == player1) {
              System.out.println("フォルド!");
              System.out.println(player2.getName() + "の勝ち！" + (total_chip -
              uncertain) + "チップ獲得！");
              player2.back(uncertain);
              player2.earn(total_chip - uncertain);
              player1.lostC();
              player2.anteLostC();
              showChip(player1, player2);
              System.out.println("");
            } else {
              System.out.println("フォルド!");
              System.out.println(player1.getName() + "の勝ち！" + (total_chip -
              uncertain) + "チップ獲得！");
              player1.back(uncertain);
```

```
                player1.earn(total_chip - uncertain);
                player2.lostC();
                player1.anteLostC();
                showChip(player1, player2);
                System.out.println("");
              }
              oneGame = true;
              break;
            } else {
              System.out.println("0か1か2を入力してください");
              System.out.println("");
            }
          }
        }

        //プレイヤー交代
        if(player == player1) {
          player = player2;
        } else {
          player = player1;
        }

        if(call2nd == true) {
          break;
        }

        if(oneGame == true) {
          break;
        }

      }
    }


    //勝敗判定
    if(oneGame == false) {
      if(finalJudge(player1, player2) == 1) {
        System.out.println(player1.getName() + "の勝ち！" + (total_chip +
        ante)/2 + "チップ獲得！");
        player1.back((total_chip + ante)/2);
        player1.earn((total_chip + ante)/2);
        player2.lostC();
        showChip(player1, player2);
        System.out.println("");
      } else if(finalJudge(player1, player2) == 2) {
        System.out.println(player2.getName() + "の勝ち！" + (total_chip +
        ante)/2 + "チップ獲得！");
```

```
            player2.back((total_chip + ante)/2);
            player2.earn((total_chip + ante)/2);
            player1.lostC();
            showChip(player1, player2);
            System.out.println("");
          } else {
            System.out.println("引き分け");
            player1.back((total_chip + ante) / 2);
            player2.back((total_chip + ante) / 2);
            showChip(player1, player2);
            System.out.println("");
          }
        }

        //リセット
        cards.organize();
        cards5.clear();
        player1.resetTotal();
        player2.resetTotal();
        player1.resetLost();
        player2.resetLost();
        player1.resetCards();
        player2.resetCards();
        ++gameCount;

        //アンティと最低ベット 5 を支払えない場合
        if (player1.getChip() < 10 || player2.getChip() < 10) {
          gameSet = true;
        }

        //ゲーム数が 10 の場合
        if(gameCount > 10) {
          gameSet = true;
        }
      }
      //勝利数をカウント
      if(player1.getChip() > player2.getChip()) {
        player1.winC();
      }

      if(player1.getChip() < player2.getChip()) {
        player2.winC();
      }

      player1.resetChip();
      player2.resetChip();
    }
```

```
        //入力をやめる
        sc.close();
        System.out.println(player1.getName() + "のワンゲーム勝利数:" +
        player1.oneWinRate() + " ゲーム勝利数:" + player1.winRate() + " 稼いだチップ数:"
        + player1.getEarn());
        System.out.println(player2.getName() + "のワンゲーム勝利数:" +
        player2.oneWinRate() + " ゲーム勝利数:" + player2.winRate() + " 稼いだチップ数:"
        + player2.getEarn());
      }

    }
```

- CardsManager クラス

```
package cal;

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

/**
 * トランプを操作するクラス
 *
 */
class CardsManager {
    // トランプ配列
    private List<Integer> cards;

    //  配列の値
    //  500 : ジョーカー
    //  101～113 : スペードの 1～13
    //  201～213 : ハートの 1～13
    //  301～313 : ダイヤの 1～13
    //  401～413 : クラブの 1～13

    // コンストラクタ
    public CardsManager()
    {
        // 配列を作成
        cards = new ArrayList<Integer>(17);

        // 初期化
        organize();
    }


    // トランプの整列
```

```
        public void organize() {

            //
            cards.clear();

            // ジョーカーを格納
            cards.add(0, 500);

            for ( int i = 1; i <= 4; ++ i ) {
                // スペードを格納(添え字:1～4)
                cards.add(i, 100 + i);
            }

            for ( int i = 1; i <= 4; ++ i ) {
                // ダイヤを格納(添え字:5～8)
                cards.add(i + 4, 300 + i);
            }

            for ( int i = 1; i <= 4; ++ i ) {
                // クラブを格納(添え字:9～12)
                cards.add(i + 8, 400 + i);
            }

            for ( int i = 1; i <= 4; ++ i ) {
                // ハートを格納(添え字:13～16)
                cards.add(i + 12, 200 + i);
            }
        }


        // トランプのシャッフル
        public void shuffle()
        {
            Collections.shuffle(cards);
        }


        // トランプの取得
        public int getCard( int num )
        {
            // num が、1～17 番目以外の場合、-1 を返す
            if ( ( 1 > num ) || ( 17 < num ) ) return -1;

            int card = cards.get(num - 1);
            cards.remove(num - 1);
            // カードを返す
            return card;
```

```
    }

    // カードを戻す
    public void returnCard(List<Integer> cards5, List<Integer> changeCard) {
        for(int i = 0; i < changeCard.size(); i++) {
            cards.add(cards5.get(changeCard.get(i) - 1));
        }
    }

}
```

- Player クラス

```
package cal;

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

/**
 * プレイヤーを表すクラス
 *
 */
public class Player {
    private String name;
    private int chip = 100; //チップ数
    private int earn = 0; //稼いだチップ
    private int lost = 0; //失ったチップ
    private int all_lost = 0; //失ったチップの合計
    private int total_bet = 0; //賭けたチップの総数
    private int hand; //役
    private int raise = 0; //レイズした回数
    private int win = 0; //ゲーム勝利数
    private int one_win = 0; //ワンゲーム勝利数
    private List<Integer> cards5; //手札

    //プレイヤーを作成
    public Player(String name) {
        setName(name);
        cards5 = new ArrayList<Integer>(5);
    }

    //プレイヤー名のセッター
    public void setName(String name) {
        this.name = name;
    }
```

```
        //プレイヤー名のゲッター
        public String getName() {
            return name;
        }

        //チップ数のゲッター
        public int getChip() {
            return chip;
        }

        //アンティをかける
        public int betAnte() {
            this.chip = chip - 5;
            this.lost += 5;
            return 5;
        }

        //手札のゲッター
        public List<Integer> getCards5() {
            return cards5;
        }

        //手札を保存
        public void setCards5(List<Integer> cards5) {
            for(int i = 0; i < cards5.size();i++) {
                this.cards5.add(cards5.get(i));
            }
        }

        //手札を更新
        public void updateCards5(List<Integer> cards5) {
            for(int i = 0; i < cards5.size();i++) {
                this.cards5.set(i, cards5.get(i));
            }
        }

        //役のゲッター
        public int getHand() {
            return hand;
        }

        //役のセッター
        public void setHand(int hand) {
            this.hand = hand;
        }

        //ベットする
```

```
    public int bet(int bet_value) {
        this.chip = chip - (bet_value - total_bet);
        this.lost += (bet_value - total_bet);
        int pre_total = total_bet;
        total_bet += (bet_value - total_bet);
        return (bet_value - pre_total);
    }

    //オールベットする
    public void allBet(int bet_value) {
        this.chip = chip - bet_value;
        this.lost += bet_value;
        total_bet += bet_value;
    }

    //賭け金の差額
    public boolean diffelence(int min) {
        if(chip > (min - total_bet)) {
            return true;
        } else {
            return false;
        }
    }

    //プレイヤーの合計ベット数を返す
    public int getTotal() {
        return total_bet;
    }

    //チップを払い戻す
    public void back(int value) {
        this.chip += value;
        this.lost -= value;
    }

    //チップを獲得する
    public void earn(int value) {
        this.chip += value;
        this.earn += value;
        ++one_win;
    }

    //失ったチップをカウントする
    public void lostC() {
        all_lost += lost;
    }
```

```
        //失ったチップをカウントする
        public void anteLostC() {
            all_lost += 5;
        }

        //レイズの回数が二回かを判断する
        public boolean raiseT() {
            if(raise == 2) {
                return true;
            } else {
                return false;
            }
        }

        //レイズの回数をカウントする
        public void raiseC() {
            ++raise;
        }


        //ワンゲームの勝利数をカウントする
        public void winC() {
            ++win;
        }

        //賭け金の合計とレイズの回数をリセット
        public void resetTotal() {
            this.total_bet = 0;
            this.raise = 0;

        }

        //手札をリセット
        public void resetCards() {
            cards5 = new ArrayList<Integer>(5);
        }

        //チップ数をリセット
        public void resetChip() {
            this.chip = 100;
        }

        //失ったチップ数をリセット
        public void resetLost() {
            this.lost = 0;
        }
```

```
        //ワンゲームの勝利数
        public int oneWinRate() {
            return one_win;
        }

        //ゲームの勝利数
        public int winRate() {
            return win;
        }

        //稼いだチップ(得た数-失った数)を返す
        public int getEarn() {
            return earn - all_lost;
        }


    }
```