卒業研究報告書

題目

Java を用いたミニ麻雀の開発

指導教員

石水 隆　講師

報告者

15-1-037-0038

田本智也

近畿大学理工学部情報学科

令和 3 年 2 月 1 日提出

概要

麻雀は古くから楽しまれてきたゲームである．麻雀には様々な役があり，和了時に完成している役により点数が入る．このため，麻雀を楽しむためには様々な役と点数計算を覚えなければならず，初心者が麻雀を学ぶ際の壁になっている．

麻雀には使用する牌の種類や有効な役が異なる様々なヴァリエーションがある．麻雀のヴァリエーションの一つがミニ麻雀である．ミニ麻雀のルールは麻雀と同じであるが通常 13 枚の手牌で行うものを 7 枚で行うので，麻雀の複雑さが緩和され，初心者でも始めやすいゲームとなっている．

本研究では，麻雀が難しくてできないといった人でも理解し楽しめるように Java を用いてミニ麻雀プログラムを作成する．

目次

1　序論

1.1　本研究の背景

麻雀は古くから広くプレイされており，時代と共に様々なローカルルールが作られている．しかし麻雀は牌の数が多いことや他の一般的なボードゲームに比べてルールが複雑であるということもあり，「牌をすべて覚えきれない」，「ルールが複雑でわからない」といった理由で手を出しづらいゲームとも言える．そこで麻雀のヴァリエーションの一つであるミニ麻雀を利用し，初心者でも始めやすく覚えやすいゲームの開発を行う．

1.2　麻雀の基本ルール[1]

麻雀は34種類の牌を各4枚ずつ,計136枚の牌を用いて3～4人でプレイするゲームである．牌の種類には萬子，筒子，索子がそれぞれ9種類，字牌が7種類存在する．各プレイヤーは13枚の牌を手牌として目前に配置し，順に山から牌を1枚引いて1枚捨てる行為を繰り返す．手牌13枚とアガリ牌1枚を合わせた計14枚を定められた形に揃えることを目指す．アガリ形の難易度に応じて点棒のやり取りが行われ，最終的に最も多くの得点を保持していたものを勝者とする．役によって点数が異なり，種類は37種にも及ぶ．また，詳しいルールについては，ミニ麻雀のルールと類似するのでここでは省略する．

1.3　麻雀のヴァリエーション[8]

麻雀には一般的なルールとは異なった様々なヴァリエーションが存在する．ここでは，それらを少しピックアップし説明する．

・三人麻雀[8]

基本的には二萬～八萬を除外した27種108枚の牌を用いて行う．北家がいない，チーができない，途中流局がない，普通のドラはすべて有効などの決まりがある．東南戦の場合は，東南ともに3局ずつ合計6局行う．

・二人麻雀[8]

基本的な部分は四人麻雀と同じ．異なる点は，北家・西家がないこと，東南戦の場合流局を数えなければ東南ともに2局ずつ合計4局行う．

・競技麻雀[8]

純粋な実力を競いやすくするために偶然性の高いリーチ時の一発，裏ドラ，槓ウラなどを排除したルールによって行われる．ただし，天和などは偶然性の役ではあるが認められている．

・アルシーアル麻雀[8]

アルシャル麻雀とも呼ばれ，20 符底の麻雀を意味する．リーチ麻雀の原型的なルールであり，現在でも日本麻雀連盟がこのルールを採用して競技を行っている．最低点のアガリ点の元が 22 符からなることからアルシーアル（中国語で 22 の意）と呼ばれる．

・ブー麻雀[8]

誰かの持ち点が倍になるか誰かの点棒がなくなった時点で終了となる．

1.4　既存の麻雀プログラム

麻雀は不確定ゲームかつ非完全情報ゲームであるため，良い手の判定が難しく，将棋や囲碁と比べると強い麻雀プログラムの作成は困難である．初期の麻雀プログラムでは，他家の手牌が見えないことを利用して他家は配牌時に一向聴となるようにし，聴牌時以外は全て自摸切りし聴牌すればリーチ，というものがよく用いられた．当然これはイカサマであり，麻雀 AI と呼べるものではない．

麻雀プログラムを作る上で比較的有望なのが，手牌および相手の捨牌から各牌の残り枚数を求め，最も和了率が高くなるように捨牌を選ぶ手法である．この手法は平均的にはそれなりの和了率が得られる一方，放銃を考えていないため高得点を振り込んでしまうことも多い．

また，昨今では，機械学習によるゲーム AI が注目されており，将棋や囲碁ではプロに勝つものも現れている．しかし，プロ棋士の棋譜を学習データとして用いることができる将棋や囲碁とは違い，適当な学習データが存在しない麻雀では機械学習も難しい．とは言え，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，ある程度の強さを持つ麻雀 AI も存在している[6][7]．

1.5　本研究の目的

本研究の背景で述べた理由からミニ麻雀の作成を目的とする，普通の麻雀では 34 種 136 枚の牌を使ってゲームを行うが，ミニ麻雀では牌は萬子をすべて抜き，残りの 25 種で行う．さらに，ミニ麻雀では，一局中に 25 種 100 枚すべての牌が使われるわけではなく，そのうちの 68 枚がランダムで使われる．また，普通の麻雀では，手牌が 13 枚で，対子 1 組と，順子，刻子，槓子のいずれかから成る面子 4 組揃うと和了となるが，ミニ麻雀では手牌が 7 枚で対子 1 組と面子 2 組がそろうと和了となる．普通の麻雀は牌が多く一般的なボードゲームと比べるとかな

り難しいが，ミニ麻雀では手牌が 7 枚なので，鳴きがなければ 9 巡目で終了となる．

このようなミニ麻雀特有の条件に基づき，プログラムを作成していく．

1.6　本報告書の構成

本報告書の構成は以下の通りである．まず，2 章においてミニ麻雀のルール・役について説明する．普通の麻雀と異なる部分についても比較しながら説明していく．続いて 3 章では本研究で取り組むミニ麻雀ゲームの開発および対 CPU 戦における戦術について説明する．また，ミニ麻雀プログラムの開発に先立ち和了形の判定プログラムの作成を行ったので，それについての説明を行う．4 章では本研究で取り組んだ判定プログラムが判定を行う事柄とプログラムに関する説明をする．5 章では作成したプログラムから得られた結果に基づき考察を述べる．6 章では今回完成に至らなかったミニ麻雀プログラムの完成に向けた今後の計画及び本研究の結論を述べる．

2　ミニ麻雀について

本章ではミニ麻雀の条件におけるルールや役について説明する．図 1，図 2 にミニ麻雀の参考画像を示す．

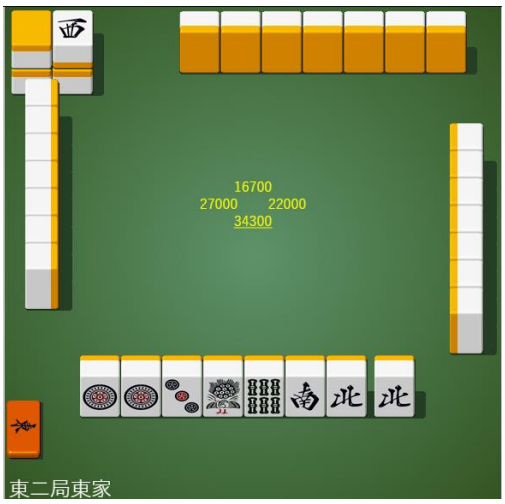

図 1

図 2

萬子抜き，手牌 7 枚で行うミニ麻雀の参考画像[13]

2.1　ミニ麻雀のルール[8]

ここではミニ麻雀の基本的なルールを列挙していく．

まず，ミニ麻雀で使用する牌については以下の通りである．

・牌は萬子をすべて抜き，筒子 9 枚，索子 9 枚，字牌 7 枚の 25 種 100 枚を使用する．

・一局中は 25 種 100 枚すべての牌が使われるわけではなく，そのうちの 68 枚がランダムで使われる．

・筒子，索子は 1～9 までの 9 種，字牌は四風牌の東南西北の 4 種と三元牌の白發中の 3 種である．

・各プレイヤーは手牌を 7 枚持ち，山から順に 1 枚引いて 1 枚捨てる行為を繰り返し行い，手牌 7 枚とアガリ牌 1 枚を合わせた計 8 枚を定められた和了形に整えることを目指す．和了形は，雀頭 1 組と面子 2 組から成る．雀頭は同じ種類の 2 枚の牌から成り，面子は順子，刻子，槓子のいずれかである．

順子，刻子，槓子は以下の条件を満たす牌の組み合わせから成る．

・順子とは，同じ種類の数牌で数が連続している 3 枚の組み合わせのことを指す．

・刻子とは，同じ牌を 3 枚揃えた組み合わせのことを指す．数牌に限らず，風牌や三元牌でも同じ牌を 3 枚揃えれば刻子を作ることができる．

・槓子とは，同じ牌を 4 枚揃えた組み合わせのことを指す．

各プレイヤーは，他家が捨牌したときに，条件を満たせば「鳴く」ことができる．鳴きには，ポン，チー，カンの 3 種類がある．また，他家の捨牌で和了形が完成するときはロンを宣言できる．以下に各鳴きができる条件挙げる．

・ポンとは同じ牌が 2 枚あるとき，ほかのプレイヤーが捨てた牌を拾って 3 枚にすることを指す．

・チーとは連続する 3 枚の順子を作る拾い方であり，ポンやカンは他の 3 人のプレイヤーが捨てた牌からできるがチーは，上家が捨てた牌からしかできない．

・カンとは手持ちに，すでに 3 枚の同じ牌があり 4 枚目を自分で引くか他家が捨てた場合に槓子を作ることである．自分で引いた牌でカンした場合は暗槓といい，鳴きにならない．ほかの人が捨てた牌をカンした場合は明槓といい，鳴きになる．

・ロンとは他の人が捨てた牌でアガル「ロンアガリ」，またはその発声を指す．他人の捨てた牌でアガル時には「ロン」と言わなければならない．

和了形が完成した場合，和了形に含まれる役により得点が決まる．ここでの役とは，和了したときの手牌の特定パターンのことである．

プレイヤーの 1 人が親となり，残りのプレイヤーは子となる．親とはその局の東家のことを指し，親は子の 1.5 倍の得点能力を有する．自摸により和了した場合，残りのプレイヤーが得点を分割して支払う．親が自摸和した場合は，得点の 0.5 倍ずつを他の 3 人の子が支払い，子が自摸和した場合は，親は得点の 1/2 を，子は得点の 1/4 を支払う．ロンにより和了した場合は，栄和したのが親ならば得点の 1.5 倍を，子ならば得点をそのままロンされた牌を捨てたプレイヤーが支払う．

実際に用いられる牌の種類について列挙する．図 3 は筒子，図 4 は索子，図 5 は字牌を表している．

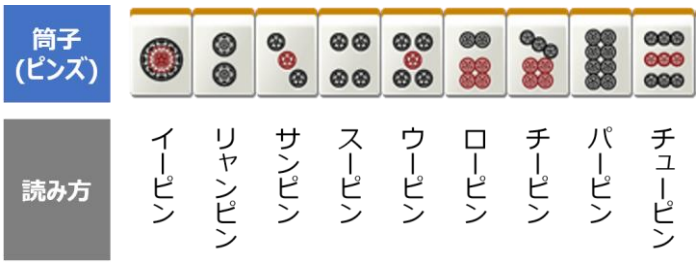

図 3　筒子

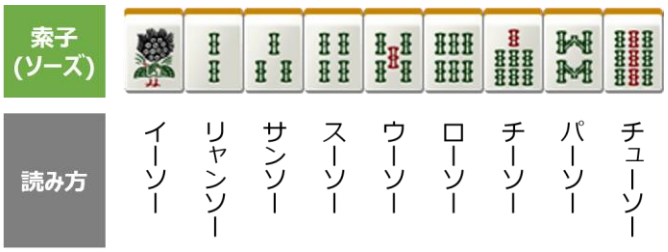

図 4　索子

図 5　字牌

2.2　ミニ麻雀の役[9][12]

普通の麻雀には様々な役があるが，ミニ麻雀では手牌が 7 枚であることや萬子を用いないといった点から平和，立直，対々和，ホンイツ，清一色，タンヤオ，役牌などの限られた役しか作ることができない．よってここではミニ麻雀の条件で成立する役とその条件及び翻数について述べる．

・平和（ピンフ）の条件
  ・すべてのメンツが順子であること．
  ・面前であること．
  ・聴牌時の待ちが両面待ちであること．
  ・頭が役のつく牌で作られていないこと．
  ・1 翻．

・立直（リーチ）
  ・聴牌であること．
  ・面前であること．
  ・自分の点数が 1000 点以上であること．
  ・自分のツモがあと 1 回以上残っていること．
  ・1 翻．

・対々和（トイトイホー）
  ・同じ牌で作った面子（刻子）のみで手牌を作り上げること．
  ・2 翻．

・混一色（ホンイーソー）
  ・字牌と筒子，字牌と索子のどちらかの組み合わせで手牌をアガリ形に持っていくこと．
  ・3 翻．

・清一色（チンイーソー）
  ・字牌を含まず，筒子，索子のどちらか 1 種類のみで手牌を整えること．
  ・6 翻．

・断么九（タンヤオ）
  ・1，9，字牌を使わずに 2～8 の数牌のみで手牌を作ること．
  ・1 翻．

・役牌（ヤクパイ）
  ・三元牌（白、發、中）のいずれかが刻子であること.
  ・場風牌が刻子であること.
  ・自風牌が刻子であること.
  ・1 翻.

・面前清自摸和（メンゼンチンツモホー，略称ツモ）
  ・門前でテンパイすること.（テンパイ形はどのような形でも良い）
  ・自身のツモによってアガルこと.
  ・1 翻.

・一盃口（イーペイコー）
  ・牌の種類も，数の並びも同じ順子を 2 組作ること.
  ・1 翻で門前でのみ成立する.
  ・1 翻.

・海底撈月（ハイテイラオユエ）
  ・その局における最後の一枚でツモアガリすること.
  ・鳴いていても成立する.
  ・1 翻.

・河底撈魚（ホウテイラオユイ）
  ・その局における最後の捨て牌をロンすることで成立する.
  ・1 翻.

・嶺上開花（リンシャンカイホウ）
  ・カンをしたときのみツモルことができる嶺上牌で上がることで成立する.
  ・聴牌時にカンをして，嶺上牌でツモしたときのみ適用される.
  ・1 翻.

・槍槓（チャンカン）
  ・他のプレイヤーが加カンをした牌が自分のアガリ牌だった場合に成立する.
  ・1 翻.

・混全帯么九（チャンタ）

・数牌の 1 か 9 と字牌をすべての面子と雀頭に含め，順子が 1 つ以上あることで成立する．

・2 翻.

・純全帯么九（ジュンチャン）

・数牌の 1 か 9 をすべての面子と雀頭に含め，順子が 1 つ以上あることで成立する．

・3 翻.

・混老頭（ホンロートー）

・1，9，字牌のみで手牌を作ること．

・必ず対々子が複合する．

・4 翻.

・緑一色（リューイーソー）

・索子の 2，3，4，6，8，と字牌の發のみで手を作ることで成立する．

・役満.

・字一色（ツーイーソー）

・字牌のみで手牌を作ることで成立する．

・役満.

・清老頭（チンロートー）

・1，9 牌のみで手牌を作ることで成立する．

・役満.

・天和（テンホウ）

・親が配牌時にすでに和了している際に成立する．

・役満.

・地和（チーホウ）

・子が配牌時点で聴牌し、かつ第一ツモで和了することで成立する．

・役満.

3　研究内容

本研究では，Java を用いてにミニ麻雀のプログラムを作成する．まずは 4 人で対人戦の行える基盤となるプログラムを作成する．その後，プレイヤーが 4 人揃わない場合にもできるようにするために，対 CPU 戦の機能を加えていく．対 CPU 戦では異なる CPU が同じような手を目指すのではなくそれぞれに戦術を持たせ，和了を目指していくものとする．現在考えている戦略は以下のとおりである．

・最短で聴牌を目指す戦略[10]

この戦略では，牌効率や牌理といったものを考慮し，最短でテンパイにたどり着くために捨て牌を選ぶ．つまり，面子を 4 つと雀頭 1 つを最短で揃えることを目指していく．具体的な方法として，孤立牌から優先的に切っていくものとする．手牌の筒子が①⑤⑦とした場合，⑤や⑦は⑥を引くことができれば面子が完成するが，①は面子を作るのに 2 牌必要になる．よって，このような手牌の場合の孤立牌は①となり，牌効率を考えた上で切る牌は①となる．

・守りに特化した戦略[11]

この戦略では，勝つことよりも負けることを回避する，あるいは負けたときに奪われる点数をいかに下げるかという点に特化し牌を選ぶ．もっとも重要な点としてはロンで点数を奪われないことが挙げられる．同じ役や点数で負けた場合でも，ロンが直撃した場合・ロンを回避した場合・ツモで負けた場合で点差が大きく変わってくる．このような最悪の場合を可能な限り避けるために以下の戦術をとる．

・捨て牌はロンできないことを利用する．つまり，手牌のどちらを切るか悩んだ場合，ほかのプレイヤーは捨て牌に多いほうを抱えて，少ないほうを切るといったもの．

・序盤の捨て牌の周辺は安全な確率が高いことや捨て牌にある数字牌のプラスマイナス 3 の数字の牌は比較的安全な確率が高いことを考慮する．

また，ミニ麻雀プログラムを作成するに先立ち，まずは和了形の判定を行うプログラムの作成を行った．麻雀を簡略化したゲームとは言え，それでも役の種類は多く点数計算も複雑であるため，和了時の牌を入力するだけで役の判定と点数の計算を行うプログラムを作成した．詳しい内容については次の章で説明する．

4　和了形の判定プログラム

4.1　和了形判定の概要

本研究では，Java を用いてミニ麻雀の役判定及び点数計算を行うプログラムを作成した．付録に本研究で作成したプログラムを示す．

作成した和了形の判定プログラムは，ミニ麻雀の環境で考えられるすべての役の判定と点数の計算を行う．判定を行うにあたり，役の種類は 2.2 で触れているので点数計算について説明を行う．

点数計算には翻数と呼ばれる点数計算の要素を用いる．この要素は和了役やドラの枚数などによって増加し，最終的に和了時の翻数によって点数が振り分けられる．以下の表 1 に翻数を用いた点数の振り分け方を示す．

翻数	子の和了点	親の和了点
1	1,000	1,500
2	2,000	2,900
3	3,900	5,800
4，5	8,000	12,000
6，7	12,000	18,000
8，9，10	16,000	24,000
11，12	24,000	36,000
13~	32,000	48,000

表 1　翻数による点数の振り分け

また，翻数のみではなく符数も用いてより細かく点数の振り分けを行う方法もあるが今回の判定プログラムには使用しないので説明は省略する．

4.2　プログラムの概要

今回作成した和了形の判定プログラムは以下の 7 つのクラスから成る．ここでは各クラスの動作に関する説明を行う．また，それぞれのクラス図を作成し，各メソッドに関する説明を行う．

[Input クラス]

和了時の牌を筒子，索子，字牌の順で入力する．牌の数が正しくない場合は改めて入力を求める．0 が入力された場合はその牌の処理をスキップし，次の牌の入力へ移る．

Input
-hai: int -num: String -skip: boolean
+Input(hai: int, num: String, skip: boolean) +input(s; int): void

・Input

Input クラスのコンストラクタ.

・input メソッド

牌数が 8 枚未満に設定し筒子，索子，字牌の順で入力を行う．それぞれの牌が入力されるたびに現在の牌数を出力する．また，0 が入力された場合はその種類の牌の入力をスキップし，次の牌の入力へ移行する.

[Sort クラス]

Input クラスで入力された牌をそれぞれ小さい順で並び変え，結果を表示する.

Sort
-num: String -array: ArrayList<Integer> -copy: ArrayList<Integer> -mahjong = new Decision(): Decision
+Sort(num: String, array: ArrayList<Integer>, copy: ArrayList<Integer>) +sort(): void +sortArray(array; ArrayList<Integer>): void

・Sort

Sort クラスのコンストラクタ.

・sort メソッド

入力された牌を小さい順に並び変える．まず，数値を文字列へ変換する．その後，Collections を用いてソートを行い牌の並び替えを行う.

・sortArray メソッド

和了形の判定後，牌を小さい順に並び変える.

[Yaku クラス]

Sort クラスで並び替えられた牌から，様々な役の判定を行う．また，得られた結果から翻数の表示を行う．

Yaku
-arrayP: ArrayList<Integer> -arrayS: ArrayList<Integer> -arrayJ: ArrayList<Integer> -arrayPn: ArrayList<Integer> -arraySn: ArrayList<Integer> -arrayJn: ArrayList<Integer> -han = 0: int -yakuman = false: boolean -reach = false: boolean -doubleReach = false: boolean -naki = false: boolean -haitei = false: boolean -houtei = false: boolean -ippatsu = false: boolean -tsumo = false: boolean -rinsyankaiho = false: boolean -tyankan = false: boolean -jihu = 0: int -bahu = 0: int -dora = 0: int -machi = 0: int -add = new Add(): Add -honroto = false: boolean
+Yaku(arrayP; ArrayList<Integer>, arrayS; ArrayList<Integer>, arrayJ; ArrayList<Integer>, arrayPn; ArrayList<Integer>, arraySn; ArrayList<Integer>, arrayJn; ArrayList<Integer>) +showHan(): void +getReach(): boolean +addFirst(): void +add(): void +calculate(): void +haiNum(): boolean

+ryuiso(): boolean
+chinitsu(): boolean
+honitsu(): boolean
+chinroto(): boolean
+honroto(): boolean
+tsuisou(): boolean
+pinhu(): boolean
+tanyao(): boolean
+haku(): boolean
+hatsu(): boolean
+tyun(): boolean
+menfon(): boolean
+chanfon(): boolean
+junchan(): boolean
+chanta(): boolean
+toitoi(): boolean
+ipeko():boolean

・Yaku

Yaku クラスのコンストラクタ.

・showHan メソッド

翻数の表示を行う.

・getYakuman メソッド

ゲッター. yakuman の値を返す.

・getReach メソッド

ゲッター. reach の値を返す.

・addFirst メソッド

役満の判定を行う前に鳴きの有無, 待ちの種類, 和了形, 自風, 場風の入力を行う.

・add メソッド

役満以外の判定を行う前に鳴きの有無, 待ちの種類, 和了形, 自風, 場風, リーチの有無, ダブルリーチの有無, ドラの数の入力を行う.

・calculate メソッド

点数計算を行うためのメソッド. Calculate オブジェクトを生成し, han, jihu, bahu の初期値をそれぞれ格納する.

・haiNum メソッド

筒子，索子，字牌それぞれのサイズを足し合わせ，その合計が 8 枚となっていることを確認する．

・ryuiso メソッド

緑一色の判定を行う．

・chinitsu メソッド

清一色の判定を行う．

・honitsu メソッド

混一色の判定を行う．

・chinroto メソッド

清老頭の判定を行う．

・honroto メソッド

混老頭の判定を行う．

・tsuisou メソッド

字一色の判定を行う．

・pinhu メソッド

平和の判定を行う．

・tanyao メソッド

断ヤオの判定を行う．

・haku メソッド

白の判定を行う．

・hatsu メソッド

發の判定を行う．

・tyun メソッド

中の判定を行う．

・menfon メソッド

門風牌の判定を行う．

・chanfon メソッド

荘風牌の判定を行う．

・junchan メソッド

純全帯么九の判定を行う．

・chanta メソッド

混全帯么九の判定を行う．

・toitoi メソッド

対々和の判定を行う．

・ipeko メソッド

一盃口の判定を行う．

[ScoreCalculation クラス]

Yaku クラスで得られた翻数の結果と自風，場風を考慮して点数の計算を行う．

ScanCalculation
-jihu = 0: int -bahu = 0: int -han = 0: int
+calculate(): void

・calculate メソッド

if 文を用いて考えうるすべての翻数において親と子の判定を行い，それぞれの条件で 1，000 点～48，000 点の振り分けを行う．

[Add クラス]

鳴き，リーチ，ダブルリーチ，自風，場風，ドラの数，アガリ形，待ちの種類，海底撈月，河底撈魚，一発，嶺上開花，槍槓の情報を入力させる．

Add
-reach = false: boolean -doubleReach = false: boolean -naki = false: boolean -haitei = false: boolean -houtei = false: boolean -ippatsu = false: boolean -tsumo = false: boolean -rinsyankaiho = false: boolean -tyankan = false: boolean -jihu = 0: int -bahu = 0: int -dora = 0: int -machi = 0: int
+naki(): void +reach(): void +jihu(): void +bahu(): void

+dora(): void +agarikata(): void +machi(): void +haitei(): void +houtei(): void +ippatsu(): void +rinsyankaiho(): void +tyankan(): void

・naki メソッド

if 文を用いて鳴きの有無を(y/n)で確認する．y が確認されると true を返す．

・reach メソッド

if 文を用いてリーチの有無を(y/n)で確認する．リーチでない場合はダブルリーチの有無の確認へ進む．それぞれにおいて y が確認されると true を返す．

・jihu メソッド

東，南，西，北をそれぞれ 1，2，3，4 として自風の情報を入力する．正しく入力が確認されると，jihu にその値を代入する．

・bahu メソッド

東，南，西，北をそれぞれ 1，2，3，4 として場風の情報を入力する．正しく入力が確認されると，bahu にその値を代入する．

・dora メソッド

ドラの数を入力し，変数 dora に入力されたドラの数を代入する．

・agarikata メソッド

ツモあがりかロンあがりかの確認を行う．if 文を用いてツモあがりを(y/n)で確認し，y の場合 true を返す．

・machi メソッド

リャンメン，タンキ，カンチャン，ペンチャン，シャボをそれぞれ 1，2，3，4，5 として待ちの種類を入力する．入力された値を machi へ代入する．

・haitei メソッド

if 文を用いて海底撈月の確認を(y/n)で確認する．y が確認されると true を返す．

・houtei メソッド

if 文を用いて河底撈魚の確認を(y/n)で確認する．y が確認されると true を返す．

・ippatsu メソッド

if 文を用いて一発の確認を(y/n)で確認する．y が確認されると true を返す．

・rinsyankaiho メソッド

if 文を用いて嶺上開花の確認を(y/n)で確認する．y が確認されると true を返す．

・tyankan メソッド

if 文を用いて槍槓の確認を(y/n)で確認する．y が確認されると true を返す．

[Decision クラス]

順子，刻子の判定を行う．また，仮の頭を設定し和了形の作成を行う．

Decision
-num = “” : String -numP: Integer -numS: Integer -jihai: Integer -hai = 0: int -FoundAtama = false: boolean -skip = false: boolean -haisu = 0: int -copy = new ArrayList<Integer>(): ArrayList<Integer> -array = new ArrayList<Integer>(): ArrayList<Integer> -arrayP = new ArrayList<Integer>(): ArrayList<Integer> -arrayS = new ArrayList<Integer>(): ArrayList<Integer> -arrayJ = new ArrayList<Integer>(): ArrayList<Integer> -arrayPn = new ArrayList<Integer>(): ArrayList<Integer> -arraySn = new ArrayList<Integer>(): ArrayList<Integer> -arrayJn = new ArrayList<Integer>(): ArrayList<Integer> -sort: Sort -i = 0: int
+getArrayP(): ArrayList<Integer> +getArrayS(): ArrayList<Integer> +getArrayJ(): ArrayList<Integer> +getArrayPn(): ArrayList<Integer> +getArraySn(): ArrayList<Integer> +getArrayJn(): ArrayList<Integer> +getArraySize(): int +getFoundAtama(): boolean +getHai(): int +getSkip(): boolean +getArray(): ArrayList<Integer>

+showArray(array: ArrayList<Integer>): void
+input(s: int): void
+sort(): void
+sortArray(array: ArrayList<Integer>): void
+syuntsu(array: ArrayList<Integer>, s: int): void
+kotsu(array: ArrayList<Integer>, s: int): void
+removeAtama(s: int): void
+make(s: int): void
+makeJ(s: int): void
+copy(array: ArrayList<Integer>, s: int): void
+re(s: int): void
+reset(): void

・getArrayP メソッド

ゲッター. arrayP の値を返す.

・getArrayS メソッド

ゲッター. arrayS の値を返す.

・getArrayJ メソッド

ゲッター. arrayJ の値を返す.

・getArrayPn メソッド

ゲッター. arrayPn の値を返す.

・getArraySn メソッド

ゲッター. arraySn の値を返す.

・getArrayJ メソッド

ゲッター. arrayJn の値を返す.

・getArraySize メソッド

ゲッター. array のサイズを返す.

・getFoundAtama メソッド

ゲッター. FoundAtama の値を返す.

・getHai メソッド

ゲッター. hai の値を返す.

・getSkip メソッド

ゲッター. skip の値を返す.

・getArray メソッド

ゲッター. array の値を返す.

・showArray メソッド

for 文を用いて ArrayList の中身を表示させる.

・input メソッド

hai, num, skip の 3 つの変数を持つ input オブジェクトを生成し，それぞれの変数で入力された値を代入する.

・sort メソッド

num, array, copy の 3 つの変数を持つ sort オブジェクトを生成し，それぞれの変数で入力された値を代入する.

・sortArray メソッド

ArrayList の並び替えを行う.

・syuntsu メソッド

入力し並び替えられた牌から順子があるどうか判定する．判定ののち削除を行う.

・kotsu メソッド

入力し並び替えられた牌から刻子があるどうか判定する．判定ののち削除を行う.

・removeAtama メソッド

頭候補があれば ArrayList から削除して残りの要素を返す．頭がなければ何もしない.

・make メソッド

変数 array の値を 3 で割った数である変数 j を作成し，その個数分 syuntsu メソッドと kotsu メソッドそれぞれを実行し和了形を作成する.

・makeJ メソッド

変数 array の値を 3 で割った数である変数 j を作成し，その個数分 kotsu メソッドを実行し和了形を作成する．字牌には順子がないため刻子のみ実行を行う.

・copy メソッド

swich 文を用いて筒子，索子，字牌それぞれのパターンで ArrayList のコピーを行う.

・re メソッド

頭候補がないあるいは間違って頭候補を選んでいた場合，削除した頭候補を元に戻す.

・reset メソッド

フィールドの値をリセットする.

[Main クラス]

Main メソッドを保持し，役の判定と点数計算を行う.

Main
+main(args: String[]): static void

・main メソッド

ミニ麻雀における和了形判定プログラムの main メソッド.

5　実行結果及び考察

図 6，図 7 には役満ではない場合の牌を入力し，それが正常にそれぞれの役に判定され，点数計算されていることがわかる．同様にして，図 8，図 9 には役満である場合の牌を入力し，それが正常に役満として判定され点数計算が行われている．また，親と子が正常に区別され点数に反映されている．

```
ミニ麻雀の和了形を判定します。
筒子の牌を入力してください。
234666
現時点での牌の数は6
並び替えた結果を表示:234666
順子を削除666
刻子を削除:
全てを削除完了
全てを削除完了
筒子一番下まで到達
234666
索子の牌を入力してください。
0
現時点での牌の数は6
字牌を入力してください。
東|南|西|北|白|発|中
1|2|3|4|5|6|7
66
現時点での牌の数は8
並び替えた結果を表示:66
仮に設定した頭は:66
頭を削除:
全てを削除完了
全てを削除完了
字牌一番下まで到達
66
```

図 6 役満ではない場合の実行結果 1/2

```
鳴きましたか？　y/n
n
待ちの種類を入力してください
リャンメン:1　タンキ:2　カンチャン:3　ペンチャン:4　シャボ:5
1
ツモあがりですか？ y/n
y
自風を入力して下さい。
東|南|西|北
1|2|3|4
1
場風を入力して下さい。
東|南|西|北
1|2|3|4
2
和了形の判定結果を表示します。
リーチしましたか？ y/n
y
ドラの数を入力してください。
3
和了形の判定結果を表示します。
混一色
リーチ
ツモ
ドラ3
8翻です。
倍満:24000点です
```

図 7 役満ではない場合の実行結果 2/2

```
ミニ麻雀の和了形を判定します。
筒子の牌を入力してください。
0
現時点での牌の数は0
索子の牌を入力してください。
234888
現時点での牌の数は6
並び替えた結果を表示:234888
順子を削除888
刻子を削除:
全てを削除完了
全てを削除完了
索子一番下まで到達
234888
字牌を入力してください。
東|南|西|北|白|発|中
1|2|3|4|5|6|7
66
現時点での牌の数は8
並び替えた結果を表示:66
仮に設定した頭は:66
頭を削除:
全てを削除完了
全てを削除完了
字牌一番下まで到達
66
```

図 8 役満の場合の実行結果 1/2

```
鳴きましたか？  y/n
n
待ちの種類を入力してください
リャンメン:1  タンキ:2  カンチャン:3  ペンチャン:4  シャボ:5
1
ツモあがりですか？ y/n
y
自風を入力して下さい。
東|南|西|北
1|2|3|4
1
場風を入力して下さい。
東|南|西|北
1|2|3|4
2
和了形の判定結果を表示します。
緑一色
役満です
48000点です。
```

図 9 役満の場合の実行結果 2/2

6　結論及び今後の研究計画

本研究ではまず，ミニ麻雀における役判定及び点数計算を行うプログラムの作成をした．結果として正常に動作し，役の判定と点数の計算を行うプログラムが完成した．ミニ麻雀プログラムに関しては完成に至っていないが今後完成に向け取り組み，今回作成した和了形判定のプログラムを生かしていきたいと考えている．具体的な研究計画は以下のとおりである．

2 月

第 1 週：今まで取り組んできた内容を踏まえ，ミニ麻雀プログラムの設計を行う．

第 2 週：麻雀ソフトや参考文献などを参考にしながらミニ麻雀プログラムを作成する．

第 3 週：4 人での対人戦を可能にする．

第 4 週：ミニ麻雀ゲームが正常に動作しているかの検証を行う．

3 月

第 1 週：対 CPU 戦に対応できるように機能を追加していく．

第 2 週：研究内容で述べた戦術の動きをする CPU を作成する．

第 3 週：戦術が正常に動作しているかの検証を行う．

第 4 週：ミニ麻雀ゲームとしての最終チェックを行う．

謝辞

本研究を行うにあたり，指導してくださった石水隆先生にはとても感謝しています．また，本研究を発表までに完成させることができなかったことを非常に申し訳なく感じています．

今後は，ご指導いただきながら取り組んできたミニ麻雀プログラムを完成に向け前向きに取り組んでいきたいと考えています．本当にありがとうございました．

参考文献

[1] 石畑恭平，コンピュータ麻雀のアルゴリズム，工学社，(2007)

[2] 石畑恭平，まうじゃん的空間，「まうじゃん for java」
http://www.amy.hi-ho.ne.jp/ishihata/maujong/

[3] ポケッタブル　本格派ミニ麻雀，永岡書店，(2007)

[4] とつげき東北．科学する麻雀．講談社．(2004)．

[5] とつげき東北．おしえて！科学する麻雀．洋泉社．(2009)．

[6] 麻雀 AI Microsoft Suphx が人間のトッププレイヤーに匹敵する成績を達成，Japan News Center，Mictosoft (2019/8/29)
https://news.microsoft.com/ja-jp/2019/08/29/190829-mahjong-ai-microsoft-suphx/

[7] とつげき東北，ASAPIN，水上直紀，マイクロソフト「麻雀 AI」の衝撃…麻雀界はここまで激変する，現代ビジネス，講談社 (2019/9/30) https://gendai.ismedia.jp/articles/-/67507

[8]麻雀のルール，Wikipedia，日本ルールのヴァリエーション
https://ja.wikipedia.org/wiki/%E9%BA%BB%E9%9B%80%E3%81%AE%E3%83%AB%E3%83%BC%E3%83%AB

[9]麻雀グッズ研究所
https://majyan-item.com/

[10]麻雀 Station
https://mj-station.net/strategy/paiefficiency1/

[11]麻雀で勝つための守りの戦術 6 選～最速雀士製造法～
https://www.bodoge-intl.com/strategy/majan-defense/

[12]キンマ Web
https://kinmaweb.jp/archives/tag/%e9%ba%bb%e9%9b%80%e5%bd%b9

[13]ミニ麻雀 - GAMEDESIGN
https://www.gamedesign.jp/games/minimahjong/

[14]野中章宏，麻雀の役判定および点数計算，卒業研究報告書，近畿大学理工学部 (2016)

付録

以下に本研究で作成したプログラムのソースコードを示す.

・Input クラス

```
package minimahjong;

import java.util.Scanner;

public class Input {
 int hai; // 牌の数
 String num; // 入力された値
 boolean skip; // false の時に、その種類の牌の処理を全てとばす
 public Input(int hai, String num, boolean skip) {
  this.hai = hai;
  this.num = num;
  this.skip = skip;
 }

 //牌を入力させる
 public void input(int s) {
  if (hai < 8) {
   switch (s) {
   case 1:
    System.out.println("ミニ麻雀の和了形を判定します。");
    System.out.println("筒子の牌を入力してください。"); break;
   case 2:
    System.out.println("索子の牌を入力してください。"); break;
   case 3:
    System.out.println("字牌を入力してください。");
    System.out.println("東|南|西|北|白|発|中");
    System.out.println("１|２|３|４|５|６|７");
    break;
   }
   Scanner scan = new Scanner(System.in);
```

```
 num = scan.next(); // 入力された値
 // 0 が入力された時処理をとばす
 if (num.charAt(0) == '0') {
  skip = true;
 }
 if (num.charAt(0) != '0') {
  hai += num.length();
 }
 System.out.println("現時点での牌の数は"+hai);

 while (hai == 1 || hai == 7 || hai > 8) {
  hai -= num.length();
  switch (s) {
  case 1:
   System.out.println("もう一度筒子の牌を入力してください。"); break;
  case 2:
   System.out.println("もう一度索子の牌を入力してください。"); break;
  case 3:
   System.out.println("もう一度字牌を入力してください。");
   System.out.println("東|南|西|北|白|発|中");
   System.out.println("１|２|３|４|５|６|７");
   break;
  }
  scan = new Scanner(System.in);
  num = scan.next(); // 入力された値
  // 0 が入力された時処理をとばす
  if (num == "0") {
   skip = true;
  }
  if (num != "0") {
   hai += num.length();
  }
  System.out.println("現時点での牌の数は"+hai);
  scan.close();
 }
}
```

```
 }
}
```

・Sort クラス

```
package minimahjong;

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

public class Sort {
 String num; // 入力された値
 ArrayList<Integer> array;
 ArrayList<Integer> copy; // 入力された値のコピー
 Decision mahjong = new Decision();

 public Sort(String num, ArrayList<Integer> array, ArrayList<Integer> copy) {
  this.num = num;
  this.array = array;
  this.copy = copy;
 }

 // 入力された牌を小さい順に並び替える
 public void sort() {
  //数字が小さい順に並び替え
  String str = String.valueOf(num); // 数値を文字列へ
  for (int i = 0; i < str.length(); i++) {
   int q = (int)str.charAt(i) - '0';
   array.add(q); // 初めて array が出てくる
  }
  Collections.sort(array); // 並び替え
  System.out.print("並び替えた結果を表示:");
  mahjong.showArray(array);
  for (int c = 0; c < array.size(); c++) {
   copy.add(c, array.get(c)); // 確認済み
  }
```

```
 }

 // あがり型を判定されたあとの牌を小さい順に並びかえる
 public void sortArray(ArrayList<Integer> array) {
  Collections.sort(array);
 }

}
```

・Yaku クラス

```
package minimahjong;

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

public class Yaku {
 ArrayList<Integer> arrayP; // あがり型と判断されたピンズの牌
 ArrayList<Integer> arrayS; // あがり型と判断されたソーズの牌
 ArrayList<Integer> arrayJ; // あがり型と判断された字牌

 ArrayList<Integer> arrayPn; // あがり型と判断されたピンズの牌
 ArrayList<Integer> arraySn; // あがり型と判断されたソーズの牌
 ArrayList<Integer> arrayJn; // あがり型と判断された字牌

 int han = 0; // ハン数
 boolean yakuman = false; // 役満かどうか
 boolean reach = false; // リーチの有無
 boolean doubleReach = false; //ダブルリーチの有無
 boolean naki = false; // 鳴きの有無
 boolean haitei = false; //海底摸月の有無
 boolean houtei = false; //河底撈魚の有無
 boolean ippatsu = false; //一発の有無
 boolean tsumo = false; //ツモの有無
 boolean rinsyankaiho = false; //嶺上開花の有無
```

```
boolean tyankan = false; //槍槓の有無
int jihu = 0;
int bahu = 0;
int dora = 0;
int machi = 0;


public Yaku(ArrayList<Integer> arrayP,ArrayList<Integer> arrayS,
  ArrayList<Integer> arrayJ,ArrayList<Integer> arrayPn,
  ArrayList<Integer> arraySn,ArrayList<Integer> arrayJn) {
 this.arrayP = arrayP;
 this.arrayS = arrayS;
 this.arrayJ = arrayJ;
 this.arrayPn = arrayPn;
 this.arraySn = arraySn;
 this.arrayJn = arrayJn;
}
// ハン数の表示
public void showHan() {
 System.out.println(han +"翻です。");
}

// ゲッター
public boolean getYakuman() {
 return yakuman;
}
public boolean getReach() {
 return reach;
}

Add add = new Add();
// 役満の判定の前に入力する
public void addFirst() {
 add.naki();
 naki = add.naki;
```

```
 add.machi();
 machi = add.machi;
 add.agarikata();
 tsumo = add.tsumo;
 if (!naki && tsumo) han++;


 add.jihu();
 add.bahu();
 jihu = add.jihu;
 bahu = add.bahu;
 System.out.println("和了形の判定結果を表示します。");
}
// 役満でない時、追加入力
public void add() {
 if (!naki) {
  add.reach();
  reach = add.reach;
  doubleReach = add.doubleReach;
  if (reach)
   han++;
  else if (doubleReach)
   han += 2;
 }

 add.dora();
 dora = add.dora;
 han += dora;
 System.out.println("和了形の判定結果を表示します。");
}
// 点数計算
public void calculate() {
 ScoreCalculation calculate = new ScoreCalculation();
 calculate.han = this.han;
 calculate.jihu = this.jihu;
 calculate.bahu = this.bahu;
```

```
 calculate.calculate();
}

// 牌の数が 8 枚あるかの確認
public boolean haiNum() {
 boolean haiNum = false;
 if (arrayP.size() + arrayS.size() + arrayJ.size() == 8)
  haiNum = true;
 return haiNum;
}

// 緑一色(リューイーソー)
public boolean ryuiso() {
 boolean ryuiso = true;
 if (!arrayP.isEmpty()) ryuiso = false;
 int h = -1;
 h = arrayJ.indexOf(new Integer(6));
 if (!arrayJ.isEmpty()) { // なにかしらの字牌が含まれている時
  if (arrayJ.size() != 2 && arrayJ.size() != 3) ryuiso = false;
  if (h == -1) ryuiso = false;
 }
 int a = -1, b = -1, c = -1, d = -1;
 a = arrayS.indexOf(new Integer(1));
 b = arrayS.indexOf(new Integer(5));
 c = arrayS.indexOf(new Integer(7));
 d = arrayS.indexOf(new Integer(9));
 if (a != -1 || b != -1 || c != -1 || d != -1) ryuiso = false;
 if (ryuiso) yakuman = true;
 return ryuiso;
}

// 清一色(チンイツ)
public boolean chinitsu() {
 boolean chinitsu = false;
 int i = 0;
 if (!arrayP.isEmpty()) {
```

```
  i++;
 }
 if (!arrayS.isEmpty()) {
  i++;
 }
 if (i == 1 && arrayJ.isEmpty())  {
  chinitsu = true;
  if (naki) han += 5;
  else han +=6;
 }
 return chinitsu;
}

// 混一色(ホンイツ)
public boolean honitsu() {
 boolean honitsu = true;
 int i = 0;
 if (!arrayP.isEmpty())
  i++;
 if (!arrayS.isEmpty())
  i++;
 if (i >= 2 || arrayJ.isEmpty())
  honitsu = false;

 if (honitsu) {
  if (naki) han += 2;
  else han += 3;
 }
 return honitsu;
}

// 清老頭(チンロートー)
public boolean chinroto() {
 boolean chinroto = true;
 for (int i = 0; i < arrayP.size(); i++) {
  if (arrayP.get(i) != 1 && arrayP.get(i) != 9)
```

```
    chinroto = false;
  }
  for (int i = 0; i < arrayS.size(); i++) {
   if (arrayS.get(i) != 1 && arrayS.get(i) != 9)
    chinroto = false;
  }
  if (!arrayJ.isEmpty()) chinroto = false;
  if (chinroto) yakuman = true;
  return chinroto;
 }

 // 混老頭(ホンロートー)
 boolean honroto = false;
 public boolean honroto() {
  honroto = true;
  for (int i = 0; i < arrayP.size(); i++) {
   if (arrayP.get(i) != 1 && arrayP.get(i) != 9)
    honroto = false;
  }
  for (int i = 0; i < arrayS.size(); i++) {
   if (arrayS.get(i) != 1 && arrayS.get(i) != 9)
    honroto = false;
  }
  if (arrayJ.isEmpty())
   honroto = false;
  if (arrayP.isEmpty() && arrayS.isEmpty())
   honroto = false;

  if (honroto) han += 2;
  return honroto;
 }

 // 字一色(ツーイーソー)
 public boolean tsuisou() {
  boolean tsuisou = false;
  if (arrayP.isEmpty() && arrayS.isEmpty()) {
```

```
  tsuisou = true;
  yakuman = true;
 }
 return tsuisou;
}

// 平和(ピンフ)
public boolean pinhu() {
 boolean pinhu = true;
 if (machi != 1) pinhu = false;
 int j = -1, b = -1, s = -1, h = -1, t = -1;
 if (!arrayJ.isEmpty()) {
  j = arrayJ.indexOf(jihu);
  b = arrayJ.indexOf(bahu);
  s = arrayJ.indexOf(5);
  h = arrayJ.indexOf(6);
  t = arrayJ.indexOf(7);
  if (j != -1 || b != -1 || s != -1 || h != -1 || t != -1)
   pinhu = false;
 }
 if (arrayPn.size() == 2 || arrayPn.size() == 5 || arrayPn.size() == 8 ||
   arrayPn.size() == 11 || arrayPn.size() == 14) {
  for (int i = 2; i < arrayPn.size() - 4; i++) {
   if (arrayPn.get(i) == arrayPn.get(i+1) &&
     arrayPn.get(i+1) == arrayPn.get(i+2))
    pinhu = false;
  }
 } else {
  for (int i = 0; i < arrayPn.size() - 2; i++) {
   if (arrayPn.get(i) == arrayPn.get(i+1) &&
     arrayPn.get(i+1) == arrayPn.get(i+2))
    pinhu = false;
  }
 }

 if (arraySn.size() == 2 || arraySn.size() == 5 || arraySn.size() == 8 ||
```

```
   arraySn.size() == 11 || arraySn.size() == 14) {
  for (int i = 2; i < arraySn.size() - 4; i++) {
   if (arraySn.get(i) == arraySn.get(i+1) &&
     arraySn.get(i+1) == arraySn.get(i+2))
    pinhu = false;
  }
 } else {
  for (int i = 0; i < arraySn.size() - 2; i++) {
   if (arraySn.get(i) == arraySn.get(i+1) &&
     arraySn.get(i+1) == arraySn.get(i+2))
    pinhu = false;
  }
 }
 if (arrayJn.size() > 2) pinhu = false;
 if (naki) pinhu = false;

 return pinhu;
}

// 断ヤオ(タンヤオ)
public boolean tanyao() {
 boolean tanyao = true;
 for (int i = 0; i < arrayP.size(); i++) {
  if (arrayP.get(i) == 1 || arrayP.get(i) == 9)
   tanyao = false;
 }
 for (int i = 0; i < arrayS.size(); i++) {
  if (arrayS.get(i) == 1 || arrayS.get(i) == 9)
   tanyao = false;
 }
 if (arrayJ.size() != 0)
  tanyao = false;
 if (tanyao == true) han++;
 return tanyao;
}
```

```
// 白(ハク)
public boolean haku() {
 boolean haku = false;;
 int f = arrayJ.indexOf(new Integer(5));
 // 白が最初に出でくる位置のインデックス
 int l = arrayJ.lastIndexOf(new Integer(5));
 // 白が最後に出でくる位置のインデックス
 if (l - f == 2) {
  haku = true;
  han++;
 }
 return haku;
}
// 發(ハツ)
public boolean hatsu() {
 boolean hatsu = false;;
 int f = arrayJ.indexOf(new Integer(6));
 // 發が最初に出でくる位置のインデックス
 int l = arrayJ.lastIndexOf(new Integer(6));
 // 發が最後に出でくる位置のインデックス
 if (l - f == 2) {
  hatsu = true;
  han++;
 }
 return hatsu;
}
// 中(チュン)
public boolean tyun() {
 boolean tyun = false;;
 int f = arrayJ.indexOf(new Integer(7));
 // 中が最初に出でくる位置のインデックス
 int l = arrayJ.lastIndexOf(new Integer(7));
 // 中が最後に出でくる位置のインデックス
 if (l - f == 2) {
  tyun = true;
  han++;
```

```
 }
 return tyun;
}
// 門風牌(メンフォンパイ)
public boolean menfon() {
 boolean menfon = false;
 int f = arrayJ.indexOf(new Integer(jihu));
 // 自風牌が最初に出でくる位置のインデックス
 int l = arrayJ.lastIndexOf(new Integer(jihu));
 // 自風牌が最後に出でくる位置のインデックス
 if (l - f == 2) {
  menfon = true;
  han++;
 }
 return menfon;
}
// 荘風牌(チャンフォンパイ)
public boolean chanfon() {
 boolean chanfon = false;
 int f = arrayJ.indexOf(new Integer(bahu));
 // 自風牌が最初に出でくる位置のインデックス
 int l = arrayJ.lastIndexOf(new Integer(bahu));
 // 自風牌が最後に出でくる位置のインデックス
 if (l - f == 2) {
  chanfon = true;
  han++;
 }
 return chanfon;
}

// 純全帯(ジュンチャン)
public boolean junchan() {
 boolean junchan = true;

 if (arrayPn.size() == 2 || arrayPn.size() == 5 || arrayPn.size() == 8 ||
   arrayPn.size() == 11 || arrayPn.size() == 14) {
```

```
 for (int i = 0; i < arrayPn.size() / 3; i++) {
  if (arrayPn.get(3*i+2) != 1 && arrayPn.get(3*i+2) != 9)
   if (arrayPn.get(3*i+3) != 1 && arrayPn.get(3*i+3) != 9)
    if (arrayPn.get(3*i+4) != 1 &&
    arrayPn.get(3*i+3) != 9)
     junchan = false;
 }
 if (arrayPn.get(0) != 1 && arrayPn.get(0) != 9)
  junchan = false;
} else if (arrayPn.size() == 3 || arrayPn.size() == 6 || arrayPn.size() == 9 ||
  arrayPn.size() == 12) {
 for (int i = 0; i < arrayPn.size() / 3; i++) {
  if (arrayPn.get(3*i) != 1 && arrayPn.get(3*i) != 9)
   if (arrayPn.get(3*i+1) != 1 && arrayPn.get(3*i+1) != 9)
    if (arrayPn.get(3*i+2) != 1 &&
    arrayPn.get(3*i+2) != 9)
     junchan = false;
 }
}

if (arraySn.size() == 2 || arraySn.size() == 5 || arraySn.size() == 8 ||
  arraySn.size() == 11 || arraySn.size() == 14) {
 for (int i = 0; i < arraySn.size() / 3; i++) {
  if (arraySn.get(3*i+2) != 1 && arraySn.get(3*i+2) != 9)
   if (arraySn.get(3*i+3) != 1 && arraySn.get(3*i+3) != 9)
    if (arraySn.get(3*i+4) != 1 &&
    arraySn.get(3*i+3) != 9)
     junchan = false;
 }
 if (arraySn.get(0) != 1 && arraySn.get(0) != 9)
  junchan = false;
} else if (arraySn.size() == 3 || arraySn.size() == 6 || arraySn.size() == 9
  || arraySn.size() == 12) {
 for (int i = 0; i < arraySn.size() / 3; i++) {
  if (arraySn.get(3*i) != 1 && arraySn.get(3*i) != 9)
   if (arraySn.get(3*i+1) != 1 && arraySn.get(3*i+1) != 9)
```

```
      if (arraySn.get(3*i+2) != 1 &&
      arraySn.get(3*i+2) != 9)
       junchan = false;
   }
  }
  if (junchan) {
   if (arrayJn.isEmpty()) {
    if (naki) han += 2;
    else han += 3;
   } else junchan = false;
  }
  return junchan;
 }
 // 混全帯(チャンタ)
 public boolean chanta() {
  boolean chanta = true;

  if (arrayPn.size() == 2 || arrayPn.size() == 5 || arrayPn.size() == 8 ||
    arrayPn.size() == 11 || arrayPn.size() == 14) {
   for (int i = 0; i < arrayPn.size() / 3; i++) {
    if (arr
```

・ScoreCalclation クラス

```
package minimahjong;

public class ScoreCalculation {
 int jihu = 0; // 自風
 int bahu = 0; // 場風
 int han = 0; // ハン数

 // 点数計算
 public void calculate() {
  if (han == 1) {
   if (jihu != 1)
    System.out.println("1000 点です");
```

```
 else
  System.out.println("1500 点です");
} else if (han == 2) {
 if (jihu != 1)
  System.out.println("2000 点です");
 else
  System.out.println("3000 点です");
} else if (han == 3) {
 if (jihu != 1)
  System.out.println("4000 点です");
 else
  System.out.println("6000 点です");
} else if (han == 4 || han == 5) {
 System.out.print("満貫:");
 if (jihu != 1)
  System.out.println("8000 点です");
 else
  System.out.println("12000 点です");
} else if (han == 6 || han == 7) {
 System.out.print("跳満:");
 if (jihu != 1)
  System.out.println("12000 点です");
 else
  System.out.println("18000 点です");
} else if (han == 8 || han == 9 || han == 10) {
 System.out.print("倍満:");
 if (jihu != 1)
  System.out.println("16000 点です");
 else
  System.out.println("24000 点です");
} else if (han == 11 || han == 12) {
 System.out.print("3 倍満:");
 if (jihu != 1)
  System.out.println("24000 点です");
 else
  System.out.println("36000 点です");
```

```
  } else if (han >= 13) {
   System.out.print("数え役満:");
   if (jihu != 1)
    System.out.println("32000 点です");
   else
    System.out.println("48000 点です");
  }
 }
}
```

・Add クラス

```
package minimahjong;
import java.util.Scanner;

public class Add {
 boolean reach = false;
 boolean doubleReach = false;
 boolean naki = false;
 boolean haitei = false;
 boolean houtei = false;
 boolean ippatsu = false;
 boolean tsumo = false;
 boolean rinsyankaiho = false;
 boolean tyankan = false;
 int jihu = 0;
 int bahu = 0;
 int dora = 0;
 int machi = 0;

 // 鳴きの有無の確認
 public void naki() {
  Scanner na = new Scanner(System.in);
  System.out.println("鳴きましたか？  y/n");
  if (na.next().charAt(0) == 'y') {
   this.naki = true;
```

```
  //na.close();
 }
}
// リーチとダブルリーチの確認
public void reach () {
 if(naki == false){
  Scanner re = new Scanner(System.in);
  System.out.println("リーチしましたか？ y/n");
  if (re.next().charAt(0) == 'y') {
   this.reach = true;
   //re.close();
  } else {
   Scanner dr = new Scanner(System.in);
   System.out.println("ダブルリーチしましたか？ y/n");
   if (dr.next().charAt(0) == 'y') {
    this.doubleReach = true;
    //dr.close();
   }
  }
 }
}
// 自風の確認
public void jihu() {
 Scanner ji = new Scanner(System.in);
 System.out.println("自風を入力して下さい。");
 System.out.println("東|南|西|北");
 System.out.println("１|２|３|４");
 int i = ji.nextInt();
 if (i <= 4)
  this.jihu = i;
 //ji.close();
}
// 場風の確認
public void bahu() {
 Scanner ba = new Scanner(System.in);
 System.out.println("場風を入力して下さい。");
```

```
 System.out.println("東|南|西|北");
 System.out.println("１|２|３|４");
 int i = ba.nextInt();
 if (i <= 4)
  this.bahu = i;
 //ba.close();
}

// ドラの入力
public void dora() {
 Scanner dr = new Scanner(System.in);
 System.out.println("ドラの数を入力してください。");
 this.dora = dr.nextInt();
 //dr.close();
}

// ツモあがりかロンあがりか
public void agarikata() {
 Scanner ag = new Scanner(System.in);
 System.out.println("ツモあがりですか？ y/n");
 if (ag.next().charAt(0) == 'y') {
  this.tsumo = true;
  //ag.close();
 }
}

// 待ちの種類
public void machi() {
 Scanner ma = new Scanner(System.in);
 System.out.println("待ちの種類を入力してください");
 System.out.println("リャンメン:1 タンキ:2 カンチャン:3 ペンチャン:4 シャボ:5");
 this.machi = ma.nextInt();
 //ma.close();
}

// 海底撈月の確認
```

```
public void haitei() {
 Scanner tei = new Scanner(System.in);
 System.out.println("海底撈月ですか？ y/n");
 if (tei.next().charAt(0) == 'y') {
  this.haitei = true;
  //tei.close();
 }
}
// 河底撈魚の確認
public void houtei() {
 Scanner tei = new Scanner(System.in);
 System.out.println("河底撈魚ですか？ y/n");
 if (tei.next().charAt(0) == 'y') {
  this.houtei = true;
  //tei.close();
 }
}
// 一発の確認
public void ippatsu() {
 Scanner ip = new Scanner(System.in);
 System.out.println("一発ですか？  y/n");
 if (ip.next().charAt(0) == 'y') {
  ippatsu = true;
  //ip.close();
 }
}
// 嶺上開花の確認
public void rinsyankaiho() {
 Scanner rin = new Scanner(System.in);
 System.out.println("嶺上開花ですか？ y/n");
 if (rin.next().charAt(0) == 'y') {
  this.rinsyankaiho = true;
  //rin.close();
 }
}
// 槍槓の確認
```

```
 public void tyankan() {
  Scanner tyan = new Scanner(System.in);
  System.out.println("槍槓ですか？ y/n");
  if (tyan.next().charAt(0) == 'y') {
   this.tyankan = true;
   //tyan.close();
  }
 }
}
```

・Decision クラス

```
package minimahjong;

import java.util.ArrayList;

public class Decision {
 String num = ""; // 入力された値
 Integer numP; // ピンズ牌
 Integer numS; // ソーズ牌
 Integer jihai; // 字牌

 int hai = 0; // 牌の数

 boolean FoundAtama = false; // 使用済み
 boolean skip = false; // 0 が入力された時処理をとばす

 int haisu = 0; // 牌の枚数
 ArrayList<Integer> copy = new ArrayList<Integer>(); // copy

 // 小さい順にソート済み
 ArrayList<Integer> array = new ArrayList<Integer>();
 ArrayList<Integer> arrayP = new ArrayList<Integer>(); // ピンズ牌
 ArrayList<Integer> arrayS = new ArrayList<Integer>(); // ソーズ牌
 ArrayList<Integer> arrayJ = new ArrayList<Integer>(); // 字牌
```

```
//  頭 + 3 + 3 + ....
ArrayList<Integer> arrayPn = new ArrayList<Integer>(); // ピンズ牌
ArrayList<Integer> arraySn = new ArrayList<Integer>(); // ソーズ牌
ArrayList<Integer> arrayJn = new ArrayList<Integer>(); // 字牌

// ゲッター
public ArrayList<Integer> getArrayP() {
 return arrayP;
}
public ArrayList<Integer> getArrayS() {
 return arrayS;
}
public ArrayList<Integer> getArrayJ() {
 return arrayJ;
}
public ArrayList<Integer> getArrayPn() {
 return arrayPn;
}
public ArrayList<Integer> getArraySn() {
 return arraySn;
}
public ArrayList<Integer> getArrayJn() {
 return arrayJn;
}
public int getArraySize() {
 return array.size();
}
public boolean getFoundAtama() {
 return FoundAtama;
}
public int getHai() {
 return hai;
}
public boolean getSkip() {
 return skip;
}
```

```
public ArrayList<Integer> getArray() {
 return array;
}

// ArrayList の中身を表示する
public void showArray(ArrayList<Integer> array) {
 for (int i = 0; i < array.size(); i++) {
  System.out.print(array.get(i));
 }
 System.out.print("¥n");
}

// 入力
public void input(int s) {
 Input input = new Input(hai, num, skip);
 input.input(s);
 hai = input.hai;
 num = input.num;
 skip = input.skip;
}
// 並び替え
Sort sort;
public void sort() {
 sort = new Sort(num, array, copy);
 sort.sort();
 num = sort.num;
 array = sort.array;
 copy = sort.copy;
}
// 並び替え(ArrayList 用)
public void sortArray(ArrayList<Integer> array) {
 sort.sortArray(array);
 array = sort.array;
}

// 順子があるかを判断する
```

```
public void syuntsu(ArrayList<Integer> array, int s) {
 Integer a = 0, b = 0, c = 0;
 int j = 0, x = 0, y = 0;

 if (array.size() >= 3) {
  a = array.get(0);
  for (int i = 0; i < array.size(); i++) {
   if (a + 1 == array.get(i)) {
    b = array.get(i);
    j++;
    x = i;
    break;
   }
  }
  for (int t = 0; t < array.size(); t++) {
   if (b + 1 == array.get(t)) {
    c = array.get(t);
    j++;
    y = t;
    break;
   }
  }
 }
 if (j == 2) {
  switch (s) {
  case 1: arrayP.add(a); break;
  case 2: arrayS.add(a); break;
  case 3: arrayJ.add(a); break;
  } switch (s) {
  case 1: arrayP.add(b); break;
  case 2: arrayS.add(b); break;
  case 3: arrayJ.add(b); break;
  } switch (s) {
  case 1: arrayP.add(c); break;
  case 2: arrayS.add(c); break;
  case 3: arrayJ.add(c); break;
```

```
  }
  System.out.print("順子を削除");
  array.remove(0);
  array.remove(x-1);
  array.remove(y-2);
  showArray(array);

 }

}
// 刻子があるかを判断する
public void kotsu(ArrayList<Integer> array, int s) {
 if (array.size() >= 3) {
  if(array.get(0) == array.get(1) && array.get(1) == array.get(2)) {
   System.out.print("刻子を削除:");
   switch (s) {
   case 1 :
    arrayP.add(array.get(0));
    arrayP.add(array.get(0));
    arrayP.add(array.get(0));
    break;
   case 2 :
    arrayS.add(array.get(0));
    arrayS.add(array.get(0));
    arrayS.add(array.get(0));
    break;
   case 3 :
    arrayJ.add(array.get(0));
    arrayJ.add(array.get(0));
    arrayJ.add(array.get(0));
    break;
   }
   array.remove(0);
   array.remove(0);
   array.remove(0);
   showArray(array);
```

```
  }
 }
}

// 頭候補があればをArrayListから削除して、残りの要素を返す
int i = 0;
public void removeAtama(int s) {
 if (array.get(i) == array.get(i+1)) { // 頭が無ければなにもしない
  // 頭候補を設定完了
  System.out.println("仮に設定した頭は:"+ array.get(i) + array.get(i+1));
  System.out.print("頭を削除:");
  switch (s) {
  case 1:
   arrayP.add(array.get(i));
   arrayP.add(array.get(i));
   break;
  case 2:
   arrayS.add(array.get(i));
   arrayS.add(array.get(i));
   break;
  case 3:
   arrayJ.add(array.get(i));
   arrayJ.add(array.get(i));
   break;
  }
  array.remove(i);
  array.remove(i);
  showArray(array);
 }
 i++;
}
// あがり型をつくる
public void make(int s) {
 int j = array.size() / 3;
 for (int i = 0; i < j; i++) {
  kotsu(array, s);
```

```
  syuntsu(array, s);
 }
}
// あがり型値をつくる(字牌専用)
public void makeJ(int s) {
 int j = array.size() / 3;
 for (int i = 0; i < j; i++) {
  kotsu(array, s);
 }
}

// ArrayList をコピーする
public void copy(ArrayList<Integer> array, int s) {
 switch (s) {
 case 1:
  for (int i = 0; i < array.size(); i++)
   arrayPn.add(i, arrayP.get(i)); break;
 case 2:
  for (int i = 0; i < array.size(); i++)
   arraySn.add(i, arrayS.get(i)); break;
 case 3:
  for (int i = 0; i < array.size(); i++)
   arrayJn.add(i, arrayJ.get(i)); break;
 }
}
// 頭候補がなかったか間違っていた時に削除した頭を元に戻す
public void re(int s) {
 switch (s) {
 case 1: arrayP.clear(); break;
 case 2: arrayS.clear(); break;
 case 3: arrayJ.clear(); break;
 }
 array.clear();
 for (int i = 0; i < copy.size(); i++) {
  array.add(i, copy.get(i));
 }
```

```
 }
 // フィールドの値をリセットする
 public void reset() {
  skip = false;
  FoundAtama = false;
  num = "";
  haisu = 0;
  copy = new ArrayList<Integer>();
  array = new ArrayList<Integer>();
  i = 0;
 }
}
```

・Main クラス

```
package minimahjong;

public class Main {
 public static void main(String[] args) {
  Decision mj = new Decision();

  // 入力された筒子の判定を行う
  mj.input(1); // キーボードから入力
  if (!mj.getSkip()) { // 0 以外が入力された時
   mj.sort(); // 並べ替え
   // 頭候補を調べて頭を削除、残りを wAtama に格納
   for (int i = 0; i < mj.getArraySize() - 1; i++) {
    mj.removeAtama(1); // 頭以外取り出し
    mj.make(1); // 頭以外であがり型をつくる
    if (mj.getArraySize() == 0) {
     System.out.print("全てを削除完了¥n");
     break;
    }
    mj.re(1);
   }
   // 頭が無かった時、あがり型をつくる
```

```
 mj.make(1);
 if (mj.getArraySize() == 0) {
  System.out.print("全てを削除完了¥n");
 }
 mj.copy(mj.getArrayP(), 1);
 mj.sortArray(mj.getArrayP());
 System.out.println("筒子一番下まで到達");
 mj.showArray(mj.getArrayP());
}
// 入力された索子の判定を行う
mj.reset();
mj.input(2); // キーボードから入力
if (!mj.getSkip()) { // 0 以外が入力された時
 mj.sort(); // 並べ替え
 // 頭候補を調べて頭を削除、残りを wAtama に格納
 for (int i = 0; i < mj.getArraySize() - 1; i++) {
  mj.removeAtama(2); // 頭以外取り出し
  mj.make(2); // 頭以外であがり型をつくる
  if (mj.getArraySize() == 0) {
   System.out.print("全てを削除完了¥n");
   break;
  }
  mj.re(2);
 }
 // 頭が無かった時、あがり型をつくる
 mj.make(2);
 if (mj.getArraySize() == 0) {
  System.out.print("全てを削除完了¥n");
 }
 mj.copy(mj.getArrayS(), 2);
 mj.sortArray(mj.getArrayS());
 System.out.println("索子一番下まで到達");
 mj.showArray(mj.getArrayS());
}
// 字牌
mj.reset();
```

```
mj.input(3); // キーボードから入力
if (!mj.getSkip()) { // 0 以外が入力された時
 mj.sort(); // 並べ替え
 // 頭候補を調べて頭を削除、残りを wAtama に格納
 for (int i = 0; i < mj.getArraySize() - 1; i++) {
  mj.removeAtama(3); // 頭以外取り出し
  mj.makeJ(3); // 頭以外であがり型をつくる
  if (mj.getArraySize() == 0) {
   System.out.print("全てを削除完了¥n");
   break;
  }
  mj.re(3);
 }
 // 頭が無かった時、あがり型をつくる
 mj.makeJ(3);
 if (mj.getArraySize() == 0) {
  System.out.print("全てを削除完了¥n");
 }
 mj.copy(mj.getArrayJ(), 3);
 mj.sortArray(mj.getArrayJ());
 System.out.println("字牌一番下まで到達");
 mj.showArray(mj.getArrayJ());
}

Yaku yaku = new Yaku(mj.getArrayP(),mj.getArrayS(),mj.getArrayJ(),
  mj.getArrayPn(),mj.getArraySn(),mj.getArrayJn());
if (mj.getHai() == 8) {
 if (yaku.haiNum()) {
  yaku.addFirst();
  if (yaku.ryuiso()) System.out.println("緑一色");
  else if (yaku.tsuisou()) System.out.println("字一色");
  else if (yaku.chinroto()) System.out.println("清老頭");
  /**
   * 役満
   */
```

```
// 役満でない時
if (!yaku.getYakuman()) {
 yaku.add();
 if (yaku.toitoi()) System.out.println("対々和");
 if (yaku.ipeko()) System.out.println("一盃口");
 if (yaku.tanyao()) System.out.println("断ヤオ");
 if (yaku.pinhu()) System.out.println("平和");
 if (yaku.chinitsu()) System.out.println("清一色");
 if (yaku.honroto()) System.out.println("混老頭");
 if (yaku.honitsu()) System.out.println("混一色");
 if (yaku.junchan()) System.out.println("純全帯");
 if (yaku.chanta()) System.out.println("全帯");
 if (yaku.haku()) System.out.println("白");
 if (yaku.hatsu()) System.out.println("發");
 if (yaku.tyun()) System.out.println("中");
 if (yaku.menfon()) System.out.println("門風牌");
 if (yaku.chanfon()) System.out.println("荘風牌");

 /**
  * ローカル役
  */

 if (yaku.reach) System.out.println("リーチ");
 else if (yaku.doubleReach) System.out.println("ダブルリーチ");
 else if (yaku.naki) System.out.println("鳴きあり");
 if (!yaku.naki && yaku.tsumo) System.out.println("ツモ");

 if (yaku.haitei) System.out.println("海底撈月");
 else if (yaku.houtei) System.out.println("河底撈月");
 if (yaku.ippatsu) System.out.println("一発");
 if (yaku.rinsyankaiho) System.out.println("嶺上開花");
 else if (yaku.tyankan) System.out.println("槍槓");
}
if (yaku.dora != 0)
 System.out.print("ドラ" + yaku.dora + "¥n");
```

```
    if (yaku.getYakuman()) { // 役満の時
     System.out.println("役満です");
     if (yaku.jihu != 1)
      System.out.println("32000 点です。");
     else
      System.out.println("48000 点です。");
    } else{
     yaku.showHan();
     yaku.calculate();
    }
   }else
    System.out.println("あがり形ではありません。");
  } else
   System.out.println("入力された牌の数が正しくないです。");

 }
}
```