卒業研究報告書

題目

簡略化した 5 五将棋の完全解析

指導教員

石水 隆　講師

報告者

19–1–037–0139

立原 太陽

近畿大学理工学部情報学科

令和 05 年 2 月 3 日提出

概要

本研究では, 後退解析を用いて 5 五将棋を簡略化したミニ将棋 (以下 4 四将棋とする) で完全解析を行った．5 五将棋は, 二人零和有限確定完全情報ゲームに分類される 1970 年に楠本茂信が発表したミニ将棋の一種であり, 5 × 5 の盤面と王, 金, 銀, 角, 飛, 歩の 6 種の駒を用いる．二人零和有限確定完全情報ゲームは双方最善手を指せば理論上は初期局面から勝敗が決定されている．5 五将棋は通常の将棋より盤面が狭く, 扱う駒数も減っているが, 局面数が 10^{56}[1] ととても大きく完全解析するには難しい．一方, 5 五将棋よりも盤面が小さく駒数も少ないミニ将棋の「どうぶつしょうぎ」や「アンパンマンはじめてしょうぎ」は, 完全解析されている [2][3]. そこで本研究では後退解析を用いて 4 四将棋の全局面の勝敗を確定する．

目次

1　序論

1.1　本研究の背景

5 五将棋は, 1970 年に楠本茂信が発表したミニ将棋の一種であり, 5 × 5 の盤面と王, 金, 銀, 角, 飛, 歩の 6 種の駒を用いる. 先手駒は一段目, 後手駒は五段目の相手陣地に入ると本将棋と同じく成ることができる. その他のルールは将棋と同じである. 図 1 に 5 五将棋の初期盤面を示す.

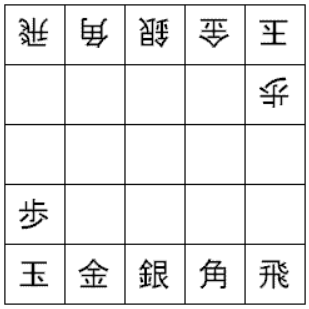

図 1　5 五将棋の初期盤面

1.2　二人零和有限確定完全情報ゲーム

5 五将棋は, 二人零和有限確定完全情報ゲームに分類される. 二人零和有限確定完全情報ゲームは, 対戦する人数が二人で, 双方の損得の合計の和が 0, 局面の数が有限で, ランダム性がなく, ゲームの情報が全て公開されているゲームのことである. 二人零和有限確定完全情報ゲームは双方最善手を指せば理論上は初期局面から勝敗は決まっている. しかし将棋や囲碁など多くのゲームでは, ゲーム中に取りうる局面の数が膨大であるためは完全解析が不可能とされている. 例を挙げれば, 可能な局面数はリバーシが 10^{28} 通り, チェスが 10^{50} 通り, 将棋が 10^{69} 通り, 囲碁が 10^{170} 通り程度あるとされており, 現在の計算機の性能を越えている. 一方, 可能な局面数が少ないゲームでは完全解析されているものもある. 連珠は双方最善手を打った場合, 47 手で先手が勝つ [7]. チェッカーは双方最善手を指すと引き分けとなる [8]. また, コネクト 4 は双方最善手を打つと 41 手で先手勝ちとなる [9].

局面数が大きいゲームについては, ゲーム盤をより小さいサイズに限定した場合の解析も行われている. サイズ 6x6 のリバーシでは, 双方最善手を打つと 16 対 20 で後手勝ちとなる [10]. また, サイズ 4x4 の囲碁は双方最善手を打つと持碁 (引き分け)[11], 5x5 の囲碁は黒の 25 目勝ちとなる [12].

将棋については, 盤面のサイズおよび駒の種類を減らしたどうぶつしょうぎやアンパンマンはじめてしょうぎでは完全解析されており, どうぶつしょうぎは双方最善手を指すと 78 手で後手勝ち [2], アンパンマンはじめてしょうぎは双方最善手を指すと引き分けとなる [3]. しかし, 5 五将棋は局面数が 10^{56}[1] ととても大きく, 現在の計算機の性能を越えているため完全解析されていない.

1.3 本研究の目的

5 五将棋は, 二人零和有限確定完全情報ゲームに分類され, 双方最善手を指せば理論上は初期局面から勝敗は決まっている. 5 五将棋は局面数が 10^{56}[1] ととても大きく完全解析するには難しい. そこで今回は 5 五将棋を簡略化したミニ将棋である 4 四将棋で完全解析を行い, 全局面の勝敗を確定する.

1.4 本報告書の構成

本報告書の構成を以下に示す. 2 章でミニ将棋について, 3 章で 4 四将棋のルール等, 4 章で今回の完全解析の研究内容の説明, 6 章で検証と考察を行い, 7 章で結論と今後の課題について述べる.

2 ミニ将棋

2.1 代表的なミニ将棋

将棋には様々なヴァリエーションがある. ヴァリエーションの中に, 通常の将棋盤よりもサイズの小さい盤面を用いるミニ将棋がある. ミニ将棋は, 盤面が小さく, また使用する駒も少ないため, 将棋の初心者のための入門用として, また, 解析用としての用途がある. 代表的なミニ将棋には, どうぶつしょうぎ, アンパンマンはじめてしょうぎ, ごろごろどうぶつしょうぎ, 5 五将棋等がある.

2.2 ミニ将棋のルール

図 2 に各ミニ将棋の初期盤面を示す.

どうぶつしょうぎは, 3 × 4 の盤面とライオン (王), ぞう, キリン, ひよこ (歩) の 4 種の駒を用いる. ぞうは斜め 4 方向 1 マスに, キリンは上下左右 1 マスに動くことができ, ひよこは相手陣地でにわとり (金) に成ることができる. 取った相手駒は自身の持ち駒にでき, 相手ライオンを取るか, 自分のライオンが相手の陣地に侵入することで勝利となる.

アンパンマンはじめてしょうぎは, 3 × 5 の盤面で行い, 先手はアンパンマン, しょくぱんまん, カレーパンマンの 3 種類の駒を, 後手はばいきんまん, ホラーマン, ドキンちゃんの 3 種類の計 6 種の駒を用いる. アンパンマン, ばいきんまんはリーダーの役割を担い, 上左右, 斜め上の 5 方向に, カレーパンマン, ドキンちゃんは上, 斜め上の 3 方向に, しょくぱんまん, ホラーマンは上左右の 3 方向に動ける. 取った相手駒は本将棋のように持ち駒にはならず, 駒の再使用はできない. 相手リーダーを取るか, 自分のリーダーが相手の陣地に侵入することで勝利となる.

ごろごろどうぶつしょうぎは, 5 × 6 の盤面とライオン (王), いぬ (金), ねこ (銀), ひよこ (歩) の 4 種の駒を用いる. 本将棋と同じく相手玉を取ることができれば勝利となる. 銀と歩は相手陣地に入れば成ることができ, 取った相手駒は自身の持ち駒にできる.

2.3 ミニ将棋の既知の結果

本節ではミニ将棋の既知の結果について述べる. どうぶつしょうぎやアンパンマンはじめてしょうぎでは完全解析されており, どうぶつしょうぎは双方最善手を指すと 78 手で後手勝ちで, 末端局面を除いた 99,485,568

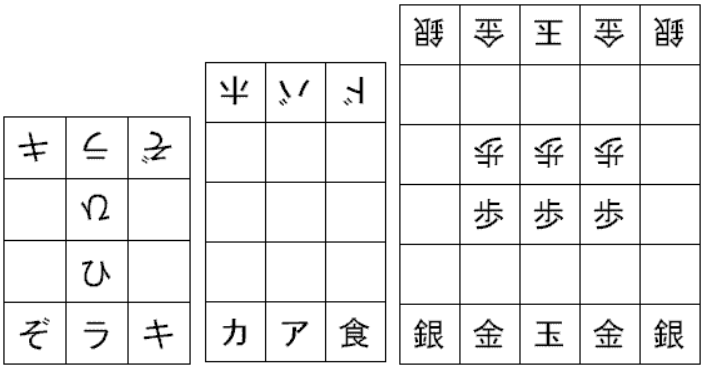

図 2　どうぶつしょうぎ, アンパンマンはじめてしょうぎ, ごろごろどうぶつしょうぎの初期局面

局面のうち, 手番の勝ちは 56,474,473 局面, 引き分けは 2,682,700 局面, 負けは 40,328,395 局面であることが分かっている [2]. アンパンマンはじめてしょうぎは双方最善手を指すと引き分けとなり, 初期局面から到達可能な 4,219,477 局面のうち, 手番の勝ちは 2,054,790 局面, 引き分けは 146,291 局面, 負けは 2,018,396 局面であることが分かっている [3].

5 五将棋は完全解析はされていないが, コンピュータ将棋の大会が開かれている. 5 五将棋の大会である UEC 杯 5 五将棋大会は, 電気通信大学 (University of Electro-Communications) エンターテイメントと認知科学研究ステーション主催で 2007 年より毎年 11 月下旬から 12 月上旬に開催されている. 5 五将棋は, 通常の将棋に比べて盤の広さが狭いので, コンピュータは既に人間よりも強くなっており, 2009 年に人間が敗れて以降, コンピュータと人間のエキシビションマッチは行われなくなっている. 普通の将棋の大会ではやりにくい試みも比較的手軽にできるという特徴がある 5 五将棋の大会は年々広がりを見せている [5]. また, 「千日手後手勝ち」というルールのもとコンピュータ対戦を行うと, 後手が勝ちやすいゲームであることが推察されている [6].

3　4 四将棋

本章では今回の研究で扱う 4 四将棋について説明する.

3.1　4 四将棋の概要

4 四将棋は, 4 × 4 の盤面を用い, 初期盤面では双方の駒は王, 金, 銀を 1 個ずつ配置する. また, 通常の将棋と異なり駒は敵陣に入っても成ることはできない. その他のルールは将棋と同じである. 4 四将棋の初期盤面を図 3 に示す.

4　4 四将棋の完全解析

4.1　局面数

4 四将棋の盤面は, 4 × 4 の 16 マスであるので, 盤上にある 1 つの駒の位置は 4 ビットで表せる. 金銀は持ち駒になる場合があるが, 玉が常に盤上にあることから, 持ち駒は自玉と同じ座標で表すことで, 金銀の位置も

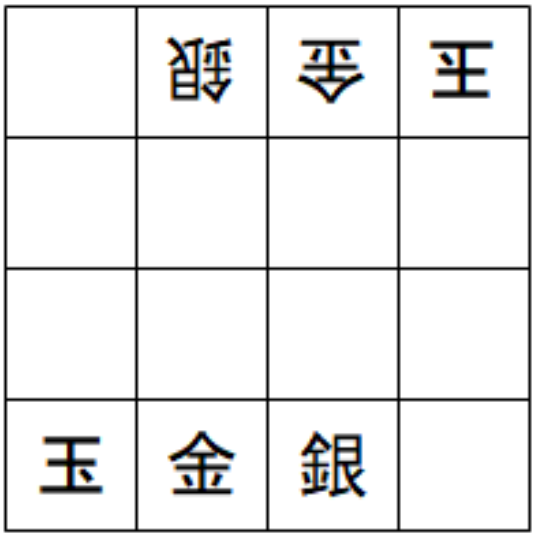

図 3　4 四将棋の初期盤面

4 ビットで表せる．金銀はそれぞれ先手駒後手駒の場合があるので各 1 ビット, 計 4 ビットで表せる．これらに先手番後手番の 1 ビットを足した計 29 ビットで全ての局面を表すことができる．よって, 到達可能性や同一局面を考慮しない場合このゲームの総局面数の上限は 2^{29} =536, 870, 912 である．完全解析が行われているどうぶつしょうぎの総局面数は 1,567,925,964 である [2] ことから, 4 四将棋の完全解析を行うことは可能であると考えられる．

4.2　後退解析

本章では, 解析法の一つである後退解析について述べる．

後退解析とは, 勝敗のついた局面から初期局面の逆方向に向かって解析していき, 全局面の勝敗を求める解析法である．後退解析を用いた完全解析の例として, 「どうぶつしょうぎ」や「アンパンマンはじめてしょうぎ」などが挙げられる [2][3].

以下に後退解析の手順を述べる．まず, 可能な全ての局面対して, その局面が勝負が付いた局面（先手勝ちまたは後手勝ち）であるかどうか判定する．次に, 全ての勝負が付いていない局面に対して, その局面から 1 手進めた局面を求め, 可能な手のうち手番側の勝ちが確定する局面へ行く手が 1 手でもあれば手番側の勝ちが確定, 全ての手が相手側の勝ちが確定する局面へ行く手であれば相手側の勝ちが確定とする．この操作を，初期局面の勝敗が確定するまで繰り返す.

後退解析は可能な全ての局面に対して勝敗を判定するため, 計算時間およびメモリ等の必要な計算資源は可能な局面の数に比例する．一方, 計算時間および計算資源の必要量は, ゲーム開始から勝負が付くまでの手数にはそれほど影響されない．このため, 可能な局面数が少ないゲームやある局面から手を進めたときに同一局面が現れるゲームの解析に向いている．

4.3　4 四将棋の後退解析

4.1 節で求めた全ての局面に対して後退解析を用いて解析する．まず, 複数の駒から王手を掛けられている局面や, 1 つのマスに複数の駒がある局面などあり得ない局面を取り除く．次に全ての局面で勝敗の確認を行い, その局面の勝敗を確定する．その後, 遷移先に手番の勝ち局面があればその局面は手番の勝ち局面, 遷移先の全てが手番の負け局面であればその局面は手番の負け局面とするという手順を全ての勝敗未確定の局面に対して繰り返し行う．この操作を全ての未確定局面に対して行ったときに, 未確定局面の数が減らなくなったら残った未確定の局面を千日手による引き分け局面とする．

5　後退解析プログラム

本章では, 後退解析に用いたプログラムについて述べる．付録に本研究で作成した後退解析プログラムのソースを示す．

5.1　State クラス

State クラスはメインの実行を行うクラスである．図 4 に State クラスのクラス図を示す．

State	メインの実行を行うクラス
+ main () : void	メインの実行を行う

図 4　State クラスのクラス図

5.2　Piece クラス

Piece クラスは駒の種類や位置などを表すクラスである．図 5 に Piece クラスのクラス図を示す．

5.3　Komadai クラス

Komadai クラスは駒台を表すクラスである．図 6 に Komadai クラスのクラス図を示す．

5.4　NextMove クラス

NextMove クラスは次の Move を表すクラスである．図 7 に NextMove クラスのクラス図を示す．

5.5　Board クラス

Board クラスは盤面を表すクラス である．図 8 に Board クラスのクラス図を示す．

5.6　Kyokumen クラス

Kyokumen クラスは局面を表すクラスである．図 9 に Kyokumen クラスのクラス図を示す．

Piece	駒の種類や位置などを表すクラス
- x : int	x 座標
- y : int	y 座標
- type : int	駒の種類
- movableFileVector, movableRankVector : int	駒の移動可能方向
+ Piece (type : int)	コンストラクタ
+ Piece (type : int, file : int, rank : int)	コンストラクタ
+ setMovableVector () : void	駒の移動可能方向をセットする
+ setInitialPosition () : void	駒を初期位置にセットする
+ setPosition (x : int, y : int) : void	駒を指定位置にセットする
+ setOnBoard () : void	駒を盤上に置く
+ removeFromBoard () : void	駒を盤上から消す
+ isOnBoard () : boolean	駒が盤上にあるか判定する
+ file () : int	駒の x 座標を返す
+ rank () : int	駒の y 座標を返す
+ type () : int	駒の種類を返す
+ name () : String	駒の名前を返す
+ move (nextFile : int, nextRank : int) : void	駒を指定位置に移動する
+ movableList (board : int[][]) : ArrayList<NextMove>	駒の移動可能な座標を返す

図 5　Piece クラスのクラス図

Komadai	駒台を表すクラス
- komadai : int[2][3]	駒台
- teban : int	手番
+ Komadai ()	コンストラクタ
+ Komadai (komadai : int[][])	コンストラクタ
+ setKomadai (i : int, teban : int) : void	駒台の駒の種類や手番を指定して一つ追加する
+ settingKomadai (komadai : int[][]) : void	駒台に駒台をセットする
+ deleateKomadai (i : int, teban : int) : void	駒台の駒の種類や手番を指定して一つ削除する
+ resetKomadai () : void	駒台をリセットする
+ getKomadai () : int[][]	駒台を得る

図 6　Komadai クラスのクラス図

NextMove	次の Move を表すクラス
- type : int	駒の種類
- nextFile : int	移動先の x 座標
- nextRank : int	移動先の y 座標
+ NextMove (type : int, nextFile : int, nextRank : int)	コンストラクタ
+ type () : int	駒の種類を返す
+ nextFile () : int	移動先の x 座標を返す
+ nextRank () : int	移動先の y 座標を返す

図 7　NextMove クラスのクラス図

Board	盤面を表すクラス
- nextFile : int	移動先の x 座標
- nextRank : int	移動先の y 座標
- gyoku : Piece	王の駒
- kin : Piece	金の駒
- gin : Piece	銀の駒
- e_gyoku : Piece	e-王の駒
- e_kin : Piece	e-金の駒
- e_gin : Piece	e-銀の駒
- movableList : ArrayList<NextMove>	候補手のリスト
- board_i : int[7]	盤面を表す 7 桁の数字
- value : int	駒を取った時の種類を表す数字
- komadai : Komadai	持ち駒用の駒台
+ Board ()	コンストラクタ
+ Board (board : int[][])	コンストラクタ
+ Board (board : int[][], komadai : int[][])	コンストラクタ
+ resetBoard () : void	盤面をリセットする
+ setKomadai (p : Piece, teban : int) : void	駒台の駒の種類や手番を指定して一つ追加する
+ settingKomadai (komadai : int[][]) : void	駒台に駒台をセットする
+ deleateKomadai (p : Piece, teban : int) : void	駒台の駒の種類や手番を指定して一つ削除する
+ resetKomadai () : void	駒台をリセットする
+ showKomadai () : void	駒台を表示する
+ setPiece (p : Piece, y : int, x : int) : void	駒を盤面の指定の位置にセットする
+ deleatePiece (p : Piece, y : int, x : int) : void	駒を盤面の指定の位置から削除する
+ emptyPiece (y : int, x : int) : void	盤面の指定の位置を空にする
+ getBoard () : int[][]	盤面を得る
+ getKomadai () : int[][]	駒台を得る
+ getMovableList () : ArrayList<NextMove>	駒の移動可能な座標を得る
+ getValue () : int	駒を取った時の種類を得る
+ showBoard () : void	盤面を表示する
+ showPiece (type : int) : String	駒を表示する
+ kinNumber () : int	盤面の金の数を返す
+ e_kinNumber () : int	盤面の e-金の数を返す
+ ginNumber () : int	盤面の銀の数を返す
+ e_ginNumber () : int	盤面の e-銀の数を返す
+ madeIndex () : int	盤面を表す 16 進数の 7 桁の数字を生成して返す
+ isMate (playerNum : int) : boolean	詰みの判定をする
+ isChecked (playerNum : int) : boolean	王手の判定をする
+ ismultyChecked (playerNum : int) : boolean	複数の駒からの王手の判定をする
+ createMovableList (playerNum : int) : void	候補手の作成をする
+ nextBoard (nextMove : NextMove, playerNum : int) : Board	候補手から次の盤面の作成をする
+ nextBoard (nextMove : NextMove, playerNum : int, komadai : int[][]) : Board	候補手から次の盤面の作成をする
+ movePiece (piece : Piece, type : int, nextFile : int, nextRank : int) : void	指定した位置に駒を移動させる
+ removePiece (nextFile : int, nextRank : int) : void	移動先の駒を取り除く

図 8　Board クラスのクラス図

Kyokumen	局面を表すクラス
- aaa : byte[536870912]	完全解析する全ての局面
- board_i : int[7]	盤面を表す 7 桁の数字
- bo : Board	盤面用の board
- teban : int	手番
- undecided, win, loss, draw, no : int	未確定, 勝ち, 負け, 引き分け, 到達不能
- gyoku, kin, gin, e_gyoku, e_kin, e_gin : Piece	王, 金, 銀, e-王, e-金, e-銀
- piece, kin1, kin2, gin1, gin2 : Piece	駒と金銀
+ Kyokumen ()	コンストラクタ
+ setkingin (board : int) : void	金銀の組み合わせをセットする
+ madeBoard (board : int) : void	盤面を表す 16 進数の 7 桁の数字から board を生成する
+ isChecked () : void	完全解析の実行をしているメソッド
+ checkMadeBoard (board : int) : boolean	Board を生成できるかどうか判定をする
+ allMotigomacheck (board : int) : boolean	金銀全てが持ち駒の場合の判定をする
+ kin1kin2gin1Check (board : int) : boolean	kin1,kin2,gin1 が持ち駒の場合の判定をする
+ kin1gin1gin2Check (board : int) : boolean	kin1,gin1,gin2 が持ち駒の場合の判定をする
+ kin1kin2gin2Check (board : int) : boolean	kin1,kin2,gin2 が持ち駒の場合の判定をする
+ kin2gin1gin2Check (board : int) : boolean	kin2,gin1,gin2 が持ち駒の場合の判定をする
+ kin1kin2Check (board : int) : boolean	kin1,kin2 が持ち駒の場合の判定をする
+ kin1gin1Check (board : int) : boolean	kin1,gin1 が持ち駒の場合の判定をする
+ kin1gin2Check (board : int) : boolean	kin1,gin2 が持ち駒の場合の判定をする
+ kin2gin1Check (board : int) : boolean	kin2,gin1 が持ち駒の場合の判定をする
+ kin2gin2Check (board : int) : boolean	kin2,gin2 が持ち駒の場合の判定をする
+ gin1gin2Check (board : int) : boolean	gin1,gin2 が持ち駒の場合の判定をする

図 9　Kyokumen クラスのクラス図

5.7　プログラムの実行方法

本節では, 本研究で作成したプログラムの実行方法を述べる. プロクラムの流れは以下の通りである.

1. サイズが 2^{29} =536, 870, 912 の配列を用意する
2. 用意した配列の全ての要素に対して 3,4 の操作を行う
3. 到達不能な局面であれば到達不能な局面を表す値にする
4. 詰みの局面であれば, 先手勝ちまたは後手勝ちを表す値にする
5. 3,4 の操作を行った後, 残りの未確定の局面に対して未確定の局面が減らなくなるまで 6 の操作を繰り返し行う
6. 未確定の次の局面をしらべて手番の勝ち局面があれば勝ち, 全てが負け局面なら負けを表す値にする
7. 未確定の局面が減らなくなると残りの未確定の局面を千日手による引き分け局面とし, 引き分けを表す値にする

上記の流れのプログラムを実行すると結果は画面に表示される. 出力された結果を図 10 に示す.

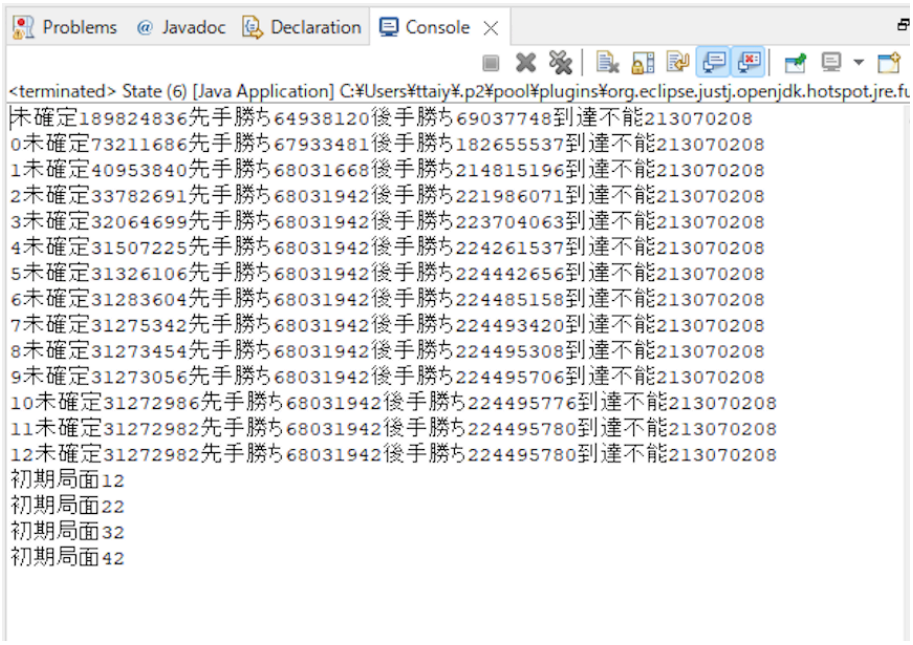

図 10　実行した結果

6　検証と考察

本研究では, 全ての局面に対して勝敗を求めた. 全ての局面の勝敗を後退解析で求めたところ, 先手勝ち局面 68,031,942, 後手勝ち局面 224,495,780, 引き分け局面 31,272,982, 到達不能局面 213,070,208 であった. ただし, この局面数には, 2 枚の金が入れ替わった局面等の同一局面も含まれている.

また, 図 3 の初期局面は後手勝ちであった. 従って, 4 四将棋は双方最善手を指すと後手必勝となる.

7　結論・今後の課題

本研究では, 5 五将棋を簡略化したミニ将棋である 4 四将棋の完全解析を行い, 後退解析を用いて双方最善手を指した時の全局面の勝敗を確定した.

今後の課題として, 4 四将棋の同一局面を除いた完全な局面数や, 初期局面からの双方最善手を指した時の手数や棋譜を求めること, まだ完全解析されていない 4 四将棋よりさらに局面数の多い二人零和有限確定完全情報ゲームの完全解析を行うことが挙げられる.

謝辞

本研究するにあたり, 適切な助言と指導をいただいた石水先生に感謝しています．ありがとうございました.

参考文献

[1] 伊藤 毅志, 新沢 剛：モンテカルロ法を用いた 5 五将棋システム, 社団法人 情報処理学会 研究報告, Vol.2007-GI-018, No.62, pp.1–6, (2007). http://id.nii.ac.jp/1001/00058488/

[2] 田中 哲郎：「どうぶつしょぎ」の完全解析, 情報処理学会研究報告, Vol.2009-GI-22 No.3, pp.1-8 (2009). http://id.nii.ac.jp/1001/00062415/

[3] 塩田 好, 石水 隆, 山本 博史：「アンパンマンはじめてしょうぎ」の完全解析, 情報処理学会関西支部 支部大会講演論文集 (2013). http://id.nii.ac.jp/1001/00096792/

[4] 高野大輔，万小紅，田中啓治，伊藤毅志：5 五将棋における認知過程の変化，情報処理学会研究報告，Vol.2011-GI-26, No.8, pp.1–8 (2011) http://id.nii.ac.jp/1001/00074627/

[5] 伊藤毅志：5 五将棋大会の動向 (2013 年〜2014 年)，情報処理学会研究報告，Vol.2015-GI-33, No.1, pp.1–5 (2015) http://id.nii.ac.jp/1001/00113628/

[6] 塩田雅弘，伊藤毅志：5 五将棋における自動対戦を用いた評価関数の学習，情報処理学会研究報告，Vol.2020-GI-44, NO.3, pp.1–6 (2020) http://id.nii.ac.jp/1001/00204861/

[7] Janos Wagner and Istvan Virag : Solving renju, ICGA Journal, Vol.24, No.1, pp.30-35 (2001), http://www.sze.hu/~gtakacs/download/wagnervirag_2001.pdf

[8] Jonathan Schaeffer, Neil Burch, Yngvi Bjorsson, Akihiro Kishimoto, Martin Muller, Robert Lake, Paul Lu, and Steve Suphen : Checkers is solved, Science Vol.317, No,5844, pp.1518-1522 (2007), http://www.sciencemag.org/content/317/5844/1518.full.pdf

[9] Victor Allis, A Knowledge-based Approach of Connect-Four, The Game is Solved: White Wins, Master Thesis, Department of Mathematics and Computer Science Vrije Universiteit (1988), http://www.informatik.uni-trier.de/~fernau/DSL0607/Masterthesis-Viergewinnt.pdf

[10] Joel Feinstein : Amenor Wins World 6x6 Championships!, Forty billion noted under the tree (July 1993), pp.6-8, British Othello Federation's newsletter., (1993), http://www.britishothello.org.uk/fbnall.pdf

[11] 清慎一, 川嶋俊：探索プログラムによる四路盤囲碁の解, 情報処理学会研究報告, GI 2000(98), pp.69–76 (2000), http://id.nii.ac.jp/1001/00058633/

[12] Eric C.D. van der Welf, H.Jaap van den Herik, and Jos W.H.M.Uiterwijk : Solving Go on Small Boards, ICGA Journal, Vol.26, No.2, pp.92-107 (2003). https://www.researchgate.net/publication/2925531_Solving_Go_On_Small_Boards/link/0fcfd511dfb651482c000000/download

付録 A　プログラムのソースコード

以下に本研究で作成した State クラスのソースコードを示す.

```
1         package syogi;
2
3   public class State {
4
5
6           public static void main(String[] args) {
7                   Kyokumen kyokumen = new Kyokumen(); // 駒の情報を登録する
8
9
10
11                  kyokumen.isChecked();
12
13
14
15
16          }
17  }
18
19
20
21
```

以下に本研究で作成した Piece クラスのソースコードを示す.

```
1   package syogi;
2
3   import java.util.ArrayList;
4
5   public class Piece {
6           //  先手の駒  盤面手前
7           final static int GYOKU = 1; // 王
8           final static int KIN = 2; // 銀
9           final static int GIN = 3; // 金
10
11          // 後手の駒 盤面奥
12          final static int E_GYOKU = -1; // 王
13          final static int E_KIN = -2; // 銀
14          final static int E_GIN = -3; // 金
15
16          final static int EMPTY = 0; // 空白
17          final static int BORDER = Integer.MAX_VALUE; // 盤外
18          boolean isOnBoard; // 駒が盤面上にあるかどうか
```

```
19          int x; // x 座標
20          int y; // y 座標
21          int type; // 駒の種類
22          int movableFileVector[], movableRankVector[]; // 移動可能方向
23
24          // 駒が移動可能な X 座標と Y 座標の方向
25          // 王{左, 左上, 上, 右上, 右, 右下, 下, 左下}
26          final static int[] GYOKU_X = {-1, -1, 0, 1, 1, 1, 0, -1};
27          final static int[] GYOKU_Y = {0, -1, -1, -1, 0, 1, 1, 1};
28          // 金  {左、左上、上、右上、右、下}
29          final static int[] KIN_X = {-1, -1, 0, 1, 1, 0};
30          final static int[] KIN_Y = {0, -1, -1, -1, 0, 1};
31          // 銀 {左上、上、右上、右下、左下}
32          final static int[] GIN_X = {-1, 0, 1, 1, -1};
33          final static int[] GIN_Y = {-1, -1, -1, 1, 1};
34
35          // 敵駒
36          final static int[] e_GYOKU_X = {-1, -1, 0, 1, 1, 1, 0, -1};
37          final static int[] e_GYOKU_Y = {0, -1, -1, -1, 0, 1, 1, 1};
38          final static int[] e_GIN_X = {1, 1, 0, -1, -1, 0};
39          final static int[] e_GIN_Y = {0, 1, 1, 1, 0, -1};
40          final static int[] e_KIN_X = {1, 0, -1, -1, 1};
41          final static int[] e_KIN_Y = {1, 1, 1, -1, -1};
42
43
44          /**
45           * コンストラクタ
46           * @param int type 駒の種類
47           */
48          public Piece(int type) {
49                  this.type = type;
50                  setMovableVector(); // 駒の移動方向を設定
51                  setInitialPosition(); // 初期位置を設定
52          }
53
54          /**
55           * コンストラクタ
56           * @param int type 駒の種類
57           * @param int file 横
58           * @param int rank  縦
59           */
60          public Piece(int type, int file, int rank) {
61                  this.type = type;
62                  setMovableVector(); // 駒の移動方向を設定
63                  this.x = file;
64                  this.y = rank;
65                  this.isOnBoard = true; // 初期は盤面上にある
66          }
67
68          /*
69           * 駒の移動可能方向をセットする
70           */
71          public void setMovableVector() {
72                  switch(type) { // 駒の種類で分岐する
73                  case GYOKU:
74                          movableFileVector = GYOKU_X;
75                          movableRankVector = GYOKU_Y;
```

```
76                      break;
77              case KIN:
78                      movableFileVector = KIN_X;
79                      movableRankVector = KIN_Y;
80                      break;
81              case GIN:
82                      movableFileVector = GIN_X;
83                      movableRankVector = GIN_Y;
84                      break;
85              case E_GYOKU:
86                      movableFileVector = e_GYOKU_X;
87                      movableRankVector = e_GYOKU_Y;
88                      break;
89              case E_KIN:
90                      movableFileVector = e_KIN_X;
91                      movableRankVector = e_KIN_Y;
92                      break;
93              case E_GIN:
94                      movableFileVector = e_GIN_X;
95                      movableRankVector = e_GIN_Y;
96                      break;
97
98              default:
99                      System.out.println("駒の種類を認識できません");
100             }
101     }
102     /*
103      * 駒を初期位置に設定する
104      * 座標は盤面の左上を (x,y)=(1,1) とする
105      */
106     public void setInitialPosition() {
107             switch(type) {
108             case GYOKU:
109                     y = 4;
110                     x = 1;
111                     break;
112             case KIN:
113                     y = 4;
114                     x = 2;
115                     break;
116             case GIN:
117                     y = 4;
118                     x = 3;
119                     break;
120             case E_GYOKU:
121                     y = 1;
122                     x = 4;
123                     break;
124             case E_KIN:
125                     y = 1;
126                     x = 3;
127                     break;
128             case E_GIN:
129                     y = 1;
130                     x = 2;
131                     break;
132             }
```

```
133                 isOnBoard = true;
134         }
135         /*
136          * 駒を指定した位置に設定する
137          */
138         public void setPosition(int x, int y) {
139                 this.x = x;
140                 this.y = y;
141                 isOnBoard = true;
142         }
143         /*
144          * 駒を盤上に置く
145          */
146         public void setOnBoard() {
147                 isOnBoard = true;
148         }
149         /*
150          * 駒を盤上から消す
151          */
152         public void removeFromBoard() {
153                 isOnBoard = false;
154         }
155         /*
156          * 駒が盤上にあるかどうかを判定する
157          */
158         public boolean isOnBoard() {
159                 return isOnBoard;
160         }
161         /*
162          * 駒の X 座標を返す
163          */
164         public int file() {
165                 return this.x;
166         }
167         /*
168          * 駒の Y 座標を返す
169          */
170         public int rank() {
171                 return this.y;
172         }
173         /*
174          * 駒の種類を返す
175          */
176         public int type() {
177                 return this.type;
178         }
179         /*
180          * 駒の名前を返す
181          */
182         public String name() {
183                 switch(type) {
184                 case GYOKU:
185                         return "王";
186                 case KIN:
187                         return "金";
188                 case GIN:
189                         return "銀";
```

```
190                 case E_GYOKU:
191                         return "王";
192                 case E_KIN:
193                         return "金";
194                 case E_GIN:
195                         return "銀";
196                 default:
197                         return "?";
198                 }
199         }
200         /*
201          * 駒を指定の位置に移動する
202          */
203         public void move(int nextFile, int nextRank) {
204                 x = nextFile;
205                 y = nextRank;
206         }
207
208         /*
209          * 駒の移動可能な座標を返す
210          */
211         public ArrayList<NextMove> movableList(int[][] board){
212                 ArrayList<NextMove> movableList = new ArrayList<NextMove>();
213                 // 移動可能な位置 (盤外を含め、6x6 として捉える)
214                 boolean[][] isMovable = {{false, false, false, false, false, false},
215                                                                 {false, true, true, true, true, false},
216                                                                 {false, true, true, true, true, false},
217                                                                 {false, true, true, true, true, false},
218                                                                 {false, true, true, true, true, false},
219                                                                 {false, false, false, false, false, false}};
220                 switch(type) { // 駒の種類による移動不可能な位置の判定
221                 case GYOKU:
222                         for(int r = 1; r <= 4; ++r) {
223                                 for(int f = 1; f <= 4; ++f) {
224                                         switch(board[r][f]) {
225                                         case GYOKU: // 自駒の位置には移動できない
226                                         case KIN:
227                                         case GIN:
228                                                 isMovable[r][f] = false;
229                                                 break;
230                                         case E_GYOKU: // 敵の王に取られる位置に移動不可
231                                                 isMovable[r][f-1] = false;
232                                                 isMovable[r+1][f-1] = false;
233                                                 isMovable[r+1][f] = false;
234                                                 isMovable[r+1][f+1] = false;
235                                                 isMovable[r][f+1] = false;
236                                                 isMovable[r-1][f+1] = false;
237                                                 isMovable[r-1][f] = false;
238                                                 isMovable[r-1][f-1] = false;
239                                                 break;
240                                         case E_KIN: // 敵の金に取られる位置に移動不可
241                                                 isMovable[r][f+1] = false;
242                                                 isMovable[r+1][f+1] = false;
243                                                 isMovable[r+1][f] = false;
244                                                 isMovable[r+1][f-1] = false;
245                                                 isMovable[r][f-1] = false;
246                                                 isMovable[r-1][f] = false;
```

```
247                                         break;
248                                 case E_GIN: // 敵の銀に取られる位置に移動不可
249                                         isMovable[r+1][f+1] = false;
250                                         isMovable[r+1][f] = false;
251                                         isMovable[r+1][f-1] = false;
252                                         isMovable[r-1][f-1] = false;
253                                         isMovable[r-1][f+1] = false;
254                                         break;
255                                 }
256                         }
257                 }
258         case KIN:
259         case GIN:
260                 for(int r = 1; r <= 4; ++r) {
261                         for(int f = 1; f <= 4; ++f) {
262                                 switch(board[r][f]) {
263                                 case GYOKU: // 自駒の位置には移動できない
264                                 case KIN:
265                                 case GIN:
266                                         isMovable[r][f] = false;
267                                         break;
268                                 }
269                         }
270                 }
271                 break;
272         case E_GYOKU:
273                 for(int r = 1; r <= 4; ++r) {
274                         for(int f = 1; f <= 4; ++f) {
275                                 switch(board[r][f]) {
276                                 case E_GYOKU: // 自駒の位置には移動できない
277                                 case E_KIN:
278                                 case E_GIN:
279                                         isMovable[r][f] = false;
280                                         break;
281                                 case GYOKU:
282                                         isMovable[r][f-1] = false;
283                                         isMovable[r-1][f-1] = false;
284                                         isMovable[r-1][f] = false;
285                                         isMovable[r-1][f+1] = false;
286                                         isMovable[r][f+1] = false;
287                                         isMovable[r+1][f+1] = false;
288                                         isMovable[r+1][f] = false;
289                                         isMovable[r+1][f-1] = false;
290                                         break;
291                                 case KIN:
292                                         isMovable[r][f-1] = false;
293                                         isMovable[r-1][f-1] = false;
294                                         isMovable[r-1][f] = false;
295                                         isMovable[r-1][f+1] = false;
296                                         isMovable[r][f+1] = false;
297                                         isMovable[r-1][f+1] = false;
298                                         isMovable[r+1][f] = false;
299                                         break;
300                                 case GIN:
301                                         isMovable[r-1][f-1] = false;
302                                         isMovable[r-1][f] = false;
303                                         isMovable[r-1][f+1] = false;
```

```
                                        isMovable[r+1][f+1] = false;
                                        isMovable[r+1][f-1] = false;
                                        break;
                                }
                        }
                }
        case E_KIN:
        case E_GIN:
                for(int r = 1; r <= 4; ++r) {
                        for(int f = 1; f <= 4; ++f) {
                                switch(board[r][f]) {
                                case E_GYOKU: // 自駒の位置には移動できない
                                case E_KIN:
                                case E_GIN:
                                        isMovable[r][f] = false;
                                        break;
                                }
                        }
                }
                break;
        }
        for(int i = 0; i < movableFileVector.length; ++i) { // 移動可能かチェックする
                int nextFile = x + movableFileVector[i];
                int nextRank = y + movableRankVector[i];
                if(isMovable[nextRank][nextFile]) {
                        NextMove nextMove = new NextMove(type, nextFile, nextRank); // 移
動可能リストに代入
                        movableList.add(nextMove);
                }
        }

        return movableList;
    }

}
```

以下に本研究で作成した Komadai クラスのソースコードを示す.

```
1  package syogi;
2
3  public class Komadai {
4
5          // 0:なし 1:金 2:銀
6
7          private int komadai[][] = new int[2][3];
8          private int teban;
9
10
11         public Komadai() {
12                 //最初は駒台に駒はない
13                 for(int i=0;i<2;i++) {
```

```
14                          for(int j=0;j<3;j++) {
15                                  komadai[i][j] = 0;
16                          }
17                  }
18          }
19
20          public Komadai(int[][] komadai) {
21                  this.komadai=komadai;
22          }
23
24          public void setKomadai(int i, int teban) {
25
26                  komadai[teban][i]+= 1;
27
28          }
29
30          public void settingKomadai(int[][] komadai) {
31                  this.komadai =komadai;
32          }
33
34          public void deleateKomadai(int i, int teban) {
35                  komadai[teban][i]-= 1;
36          }
37          public void resetKomadai() {
38
39                  for(int i=0;i<2;i++) {
40                          for(int j=0;j<3;j++) {
41                                  komadai[i][j] = 0;
42                          }
43                  }
44
45          }
46
47
48          public int[][] getKomadai(){
49                  return komadai;
50
51          }
52  }
53
```

以下に本研究で作成した NextMove クラスのソースコードを示す.

```
1  package syogi;
2
3  public class NextMove {
4          //  先手の駒  盤面手前
5          final static int GYOKU = 1; // 王
6          final static int KIN = 2; // 金
7          final static int GIN = 3; // 銀
8
9          // 後手の駒 盤面奥
10         final static int E_GYOKU = -1; // 王
11         final static int E_KIN = -2; // 金
12         final static int E_GIN = -3; // 銀
13
14         final static boolean isChessStyleScore = false; // 棋譜表記がチェスか将
   棋かどうか
15
16         int type; // 駒の種類
17         int nextFile; // 移動先の X 座標
18         int nextRank; // 移動先の Y 座標
19         int value; // 移動した場合の盤面の評価値
20
21         /**
22          * コンストラクタ
23          * @param int type 駒の種類
24          * @param int nextFile  移動先の X 座標
25          * @param int nextRank  移動先の Y 座標
26          */
27         public NextMove(int type, int nextFile, int nextRank) {
28                 this.type = type;
29                 this.nextFile = nextFile;
30                 this.nextRank = nextRank;
31         }
32
33         /**
34          * 駒の種類を返す
35          * @return  駒の種類
```

```
36              */
37             public int type() {
38                     return type;
39             }
40
41             /**
42              * 移動先の X 座標を返す
43              * @return   移動先の X 座標
44              */
45             public int nextFile() {
46                     return nextFile;
47             }
48
49             /**
50              * 移動先の Y 座標を返す
51              * @return   移動先の Y 座標
52              */
53             public int nextRank() {
54                     return nextRank;
55             }
56     }
57
```

以下に本研究で作成した Board クラスのソースコードを示す.

```
1   package syogi;
2
3   import java.util.ArrayList;
4
5   public class Board {
6           //  先手の駒  盤面手前
7           final static int GYOKU = 1; // 王
8           final static int KIN = 2; // 金
9           final static int GIN = 3; // 銀
10
11          // 後手の駒 盤面奥
12          final static int E_GYOKU = -1; // 王
13          final static int E_KIN = -2; // 金
14          final static int E_GIN = -3; // 銀
15
16          final static int EMPTY = 0; // 空白
17          final static int BORDER = Integer.MAX_VALUE; // 盤外
18          final static boolean isChessStyleScore = false; // 棋譜表記をチェス式か将棋式か
19
20          // 将棋盤面
21          public int[][] board = {
22                          {BORDER, BORDER, BORDER, BORDER, BORDER, BORDER},
23                          {BORDER, EMPTY, E_GIN, E_KIN, E_GYOKU, BORDER},
24                          {BORDER, EMPTY, EMPTY, EMPTY, EMPTY, BORDER},
25                          {BORDER, EMPTY, EMPTY, EMPTY, EMPTY, BORDER},
26                          {BORDER, GYOKU, KIN, GIN, EMPTY, BORDER},
27                          {BORDER, BORDER, BORDER, BORDER, BORDER, BORDER}
28          };
29          int nextFile; // 移動先の X 座標
30          int nextRank; // 移動先の Y 座標
31
```

```
32          // 駒
33          Piece gyoku, kin, gin, e_gyoku, e_kin, e_gin;
34          ArrayList<NextMove> movableList; // 候補手のリスト
35
36          private int[] board_i = new int [7]; // 盤面の要素
37          private int value; // 駒を取った時の種類を表す要素
38
39
40          Komadai komadai; // 持ち駒用の駒台の生成
41          /*
42           * コンストラクタ
43           *  駒の初期設定
44           */
45          public Board() {
46                  // 駒のインスタンス生成
47                  gyoku = new Piece(GYOKU);
48                  kin = new Piece(KIN);
49                  gin = new Piece(GIN);
50
51                  e_gyoku = new Piece(E_GYOKU);
52                  e_kin = new Piece(E_KIN);
53                  e_gin = new Piece(E_GIN);
54                  // 駒台を生成
55                  komadai = new Komadai();
56          }
57
58          /**
59           * コンストラクタ
60           * 盤面に駒を配置
61           * @param int[][] board 盤面
62           */
63          public Board(int[][] board) {
64                  // 盤面を更新する
65                  for(int i = 1; i <= 4; ++i) {
66                          for(int j = 1; j <= 4; ++j) {
67                                  this.board[i][j] = board[i][j];
68                          }
69                  }
70                  // 盤面にある駒を再生成する
71                  for(int i = 1; i <= 4; ++i) {
72                          for(int j = 1; j <= 4; ++j) {
73                                  switch(board[i][j]) {
74                                  case GYOKU:
75                                          gyoku = new Piece(GYOKU, j, i);
76                                          break;
77                                  case KIN:
78                                          kin = new Piece(KIN, j, i);
79                                          break;
80                                  case GIN:
81                                          gin = new Piece(GIN, j, i);
82                                          break;
83                                  case E_GYOKU:
84                                          e_gyoku = new Piece(E_GYOKU, j, i);
85                                          break;
86                                  case E_KIN:
87                                          e_kin = new Piece(E_KIN, j, i);
88                                          break;
89                                  case E_GIN:
90                                          e_gin = new Piece(E_GIN, j, i);
91                                          break;
92                                  }
93                          }
94                  }
95                  // 駒台を生成
96                  komadai = new Komadai();
97          }
98
99          /**
100          * コンストラクタ
101          * 盤面に駒を配置 駒台に駒を追加
102          * @param int[][] board 盤面
103          */
104         public Board(int[][] board, int[][] komadai) {
105                 // 盤面を更新する
106                 for(int i = 1; i <= 4; ++i) {
107                         for(int j = 1; j <= 4; ++j) {
108                                 this.board[i][j] = board[i][j];
```

```
109                     }
110             }
111             // 盤面にある駒を再生成する
112             for(int i = 1; i <= 4; ++i) {
113                     for(int j = 1; j <= 4; ++j) {
114                             switch(board[i][j]) {
115                             case GYOKU:
116                                     gyoku = new Piece(GYOKU, j, i);
117                                     break;
118                             case KIN:
119                                     kin = new Piece(KIN, j, i);
120                                     break;
121                             case GIN:
122                                     gin = new Piece(GIN, j, i);
123                                     break;
124                             case E_GYOKU:
125                                     e_gyoku = new Piece(E_GYOKU, j, i);
126                                     break;
127                             case E_KIN:
128                                     e_kin = new Piece(E_KIN, j, i);
129                                     break;
130                             case E_GIN:
131                                     e_gin = new Piece(E_GIN, j, i);
132                                     break;
133                             }
134                     }
135             }
136             // 駒台を生成
137             this.komadai = new Komadai(komadai);
138     }
139
140     public void resetBoard() {
141             for(int i = 1; i <= 4; ++i) {
142                     for(int j = 1; j <= 4; ++j) {
143                             this.board[i][j] = EMPTY;
144                     }
145             }
146     }
147
148     /**
149      *
150      * @param p
151      * @param teban 先手 0 後手 1
152      */
153     public void setKomadai(Piece p,int teban) {
154             int i = 0;
155
156             if (p.type() == KIN) {
157                     i=1;
158             }else if (p.type() == GIN) {
159                     i=2;
160             }
161
162
163             komadai.setKomadai(i, teban);
164
165     }
166
167     public void settingKomadai(int[][] komadai) {
168             this.komadai.settingKomadai(komadai);
169     }
170
171     public void deleateKomadai(Piece p,int teban) {
172             int i = 0;
173
174             if (p.type() == KIN) {
175                     i=1;
176             }else if (p.type() == GIN) {
177                     i=2;
178             }
179
180
181             komadai.deleateKomadai(i, teban);
182
183     }
184     public void resetKomadai() {
185
186             komadai.resetKomadai();
```

```
187
188         }
189         public void showKomadai() {
190                 System.out.print("先手");
191                 for(int j = 0; j < 3; ++j) {
192                         System.out.print(getKomadai()[0][j]);
193                 }
194                 System.out.print("後手");
195                 for(int j = 0; j < 3; ++j) {
196                         System.out.print(getKomadai()[1][j]);
197                 }
198
199
200
201                 System.out.println();
202
203         }
204         public void setPiece(Piece p, int y, int x) {
205                 this.board[y][x] = p.type;
206                 p.setOnBoard();
207
208         }
209
210         public void deleatePiece(Piece p, int y, int x) {
211                 if(this.board[y][x] == p.type) {
212                         this.board[y][x] = EMPTY;
213                         p.removeFromBoard();
214                 }
215
216         }
217
218         public void emptyPiece(int y, int x) {
219                 this.board[y][x] = EMPTY;
220
221         }
222
223
224         public int[][] getBorad() {
225                 return this.board;
226         }
227
228         public int[][] getKomadai() {
229                 return this.komadai.getKomadai();
230         }
231
232         public ArrayList<NextMove> getMovableList(){
233                 return movableList;
234         }
235         public int getValue() {
236                 return value;
237         }
238
239         /**
240          * 盤面を表示する
241          */
242         public void showBoard() {
243                 System.out.println("   1 2 3 4 ");
244                 for(int i = 0; i < board.length; ++i) {
245                         switch(i) {
246                         case 0:
247                                 System.out.print("  ");
248                                 break;
249                         case 1:
250                                 System.out.print("一");
251                                 break;
252                         case 2:
253                                 System.out.print("二");
254                                 break;
255                         case 3:
256                                 System.out.print("三");
257                                 break;
258                         case 4:
259                                 System.out.print("四");
260                                 break;
261                         case 5:
262                                 System.out.print("  ");
263                                 break;
```

```
264                     }
265                     for(int j = 0; j < board[i].length; ++j) {
266                             System.out.print(showPiece(board[i][j]));
267                     }
268                     System.out.println();
269             }
270     }
271
272     /**
273      * 駒を表示する
274      * @param type 駒の種類
275      * @return  駒の文字列
276      */
277     public String showPiece(int type) {
278             switch(type) {
279             case GYOKU:
280                     return "王";
281             case KIN:
282                     return "金";
283             case GIN:
284                     return "銀";
285
286             case E_GYOKU:
287                     return "お";
288             case E_KIN:
289                     return "き";
290             case E_GIN:
291                     return "ぎ";
292             case EMPTY:
293                     return "  ";
294             case BORDER:
295                     return " ■";
296             default:
297                     return "?";
298             }
299     }
300
301     /**
302      * 盤面の金の数
303      * @return
304      */
305     public int kinNumber() {
306             int x=0;
307             for(int i = 0; i < 4; ++i) {
308                     for(int j = 0; j < 4; ++j) {
309                             if(board[i+1][j+1] == KIN) {
310                                     x++;
311                             }
312                     }
313             }
314             return x;
315     }
316     /**
317      * 盤面の e_金の数
318      * @return
319      */
320     public int e_kinNumber() {
321             int x=0;
322             for(int i = 0; i < 4; ++i) {
323                     for(int j = 0; j < 4; ++j) {
324                             if(board[i+1][j+1] == E_KIN) {
325                                     x++;
326                             }
327                     }
328             }
329             return x;
330     }
331     /**
332      * 盤面の銀の数
333      * @return
334      */
335     public int ginNumber() {
336             int x=0;
337             for(int i = 0; i < 4; ++i) {
338                     for(int j = 0; j < 4; ++j) {
339                             if(board[i+1][j+1] == GIN) {
340                                     x++;
341                             }
```

```
342                     }
343             }
344             return x;
345     }
346     /**
347      * 盤面の e_銀の数
348      * @return
349      */
350     public int e_ginNumber() {
351             int x=0;
352             for(int i = 0; i < 4; ++i) {
353                     for(int j = 0; j < 4; ++j) {
354                             if(board[i+1][j+1] == E_GIN) {
355                                     x++;
356                             }
357                     }
358             }
359             return x;
360     }
361     /**
362      *
363      * board から盤面を表す 16 進数の 7 桁の数字を返す
364      */
365     public int madeIndex() {
366             // 王の位置で生成
367             for(int i =1;i<5;i++) {
368                     if(gyoku.file() == i && gyoku.rank() == 1) {
369                             board_i[0] = i-1;
370                     }
371             }
372             for(int i = 1;i<5;i++) {
373                     if(gyoku.file() == i && gyoku.rank() == 2) {
374                             board_i[0] = i+3;
375                     }
376             }
377             for(int i =1;i<5;i++) {
378                     if(gyoku.file() == i && gyoku.rank() == 3) {
379                             board_i[0] = i+7;
380                     }
381             }
382             for(int i =1;i<5;i++) {
383                     if(gyoku.file() == i && gyoku.rank() == 4) {
384                             board_i[0] = i+11;
385                     }
386             }
387             // e-王の位置で生成
388             for(int i =1;i<5;i++) {
389                     if(e_gyoku.file() == i && e_gyoku.rank() == 1) {
390                             board_i[1] = i-1;
391                     }
392             }
393             for(int i = 1;i<5;i++) {
394                     if(e_gyoku.file() == i && e_gyoku.rank() == 2) {
395                             board_i[1] = i+3;
396                     }
397             }
398             for(int i =1;i<5;i++) {
399                     if(e_gyoku.file() == i && e_gyoku.rank() == 3) {
400                             board_i[1] = i+7;
401                     }
402             }
403             for(int i =1;i<5;i++) {
404                     if(e_gyoku.file() == i && e_gyoku.rank() == 4) {
405                             board_i[1] = i+11;
406                     }
407             }
408             String kin ="00"; // 金の部分の先手後手駒の判定
409             String gin ="00"; // 銀の部分の先手後手駒の判定
410
411             // kin が 2 枚のとき
412             if(kinNumber()== 2) {
413                     int k=0;
414                     for(int i =1;i<5;i++) {
415                             for(int j =1;j<5;j++) {
416                                     if(board[i][j] == KIN) {
417                                             if(k==0) {
418                                                     board_i[2] = (i-1)*4 + j-1;
419                                             }if(k==1) {
```

```
                                        board_i[3] = (i-1)*4 + j-1;
                                }

                                k++;
                        }

                }
        }
        kin = "00";
}

// e_kin が 2 枚のとき
if(e_kinNumber()== 2) {
        int k=0;
        for(int i =1;i<5;i++) {
                for(int j =1;j<5;j++) {
                        if(board[i][j] == E_KIN) {
                                if(k==0) {
                                        board_i[2] = (i-1)*4 + j-1;
                                }if(k==1) {
                                        board_i[3] = (i-1)*4 + j-1;
                                }

                                k++;
                        }

                }
        }
        kin = "11";

}
// e_kin, kin が 1 枚のとき
if(e_kinNumber()== 1 && e_kinNumber()== 1) {
        for(int i =1;i<5;i++) {
                for(int j =1;j<5;j++) {
                        if(board[i][j] == KIN) {
                                board_i[2] = (i-1)*4 + j-1;
                        }
                        if(board[i][j] == E_KIN) {
                                board_i[3] = (i-1)*4 + j-1;
                        }
                }
        }
        kin = "01";
}

// kin が 1 枚のとき
if(kinNumber()== 1 && e_kinNumber()== 0) {
        for(int i =1;i<5;i++) {
                for(int j =1;j<5;j++) {
                        if(board[i][j] == KIN) {
                                board_i[2] = (i-1)*4 + j-1;
                        }
                }
        }
        kin = "00";
}
// e_kin が 1 枚のとき
if(kinNumber()== 0 && e_kinNumber()== 1) {
        for(int i =1;i<5;i++) {
                for(int j =1;j<5;j++) {
                        if(board[i][j] == E_KIN) {
                                board_i[2] = (i-1)*4 + j-1;
                        }
                }
        }
        kin = "10";
}
// gin が 2 枚のとき
if(ginNumber()== 2) {
        int k=0;
        for(int i =1;i<5;i++) {
                for(int j =1;j<5;j++) {
                        if(board[i][j] == GIN) {
                                if(k==0) {
                                        board_i[4] = (i-1)*4 + j-1;
                                }if(k==1) {
                                        board_i[5] = (i-1)*4 + j-1;
```

```
498                                     }
499
500                                     k++;
501                             }
502
503                     }
504             }
505             gin = "00";
506     }
507
508     // e_gin が 2 枚のとき
509     if(e_ginNumber()== 2) {
510             int k=0;
511             for(int i =1;i<5;i++) {
512                     for(int j =1;j<5;j++) {
513                             if(board[i][j] == E_KIN) {
514                                     if(k==0) {
515                                             board_i[4] = (i-1)*4 + j-1;
516                                     }if(k==1) {
517                                             board_i[5] = (i-1)*4 + j-1;
518                                     }
519
520                                     k++;
521                             }
522
523                     }
524             }
525             gin = "11";
526
527     }
528     // e_gin, gin が 1 枚のとき
529     if(e_ginNumber()== 1 && e_ginNumber()== 1) {
530             for(int i =1;i<5;i++) {
531                     for(int j =1;j<5;j++) {
532                             if(board[i][j] == GIN) {
533                                     board_i[4] = (i-1)*4 + j-1;
534                             }
535                             if(board[i][j] == E_GIN) {
536                                     board_i[5] = (i-1)*4 + j-1;
537                             }
538                     }
539             }
540             gin = "01";
541     }
542
543     // gin が 1 枚のとき
544     if(ginNumber()== 1 && e_ginNumber()== 0) {
545             for(int i =1;i<5;i++) {
546                     for(int j =1;j<5;j++) {
547                             if(board[i][j] == GIN) {
548                                     board_i[4] = (i-1)*4 + j-1;
549                             }
550                     }
551             }
552             gin = "00";
553     }
554     // e_gin が 1 枚のとき
555     if(ginNumber()== 0 && e_ginNumber()== 1) {
556             for(int i =1;i<5;i++) {
557                     for(int j =1;j<5;j++) {
558                             if(board[i][j] == E_GIN) {
559                                     board_i[4] = (i-1)*4 + j-1;
560                             }
561                     }
562             }
563             gin = "10";
564     }
565
566
567     String kingin = kin+gin;
568     //              System.out.println(kingin);
569
570     int kingin1 = Integer.parseInt(kingin, 2);
571     //              System.out.println(kingin1);
572
573     // kin e-kin
574     if(getKomadai()[0][1] == 1 && getKomadai()[1][1] == 1) {
```

```
575                         board_i[2] = board_i[0];
576                         board_i[3] = board_i[1];
577
578                 }else if(getKomadai()[0][1] == 1) {
579                         board_i[3] = board_i[0];
580                 }else if(getKomadai()[1][1] == 1) {
581                         board_i[3] = board_i[1];
582                 }
583
584                 // gin e-gin
585                 if(getKomadai()[0][2] == 1 && getKomadai()[1][2] == 1) {
586                         board_i[4] = board_i[0];
587                         board_i[5] = board_i[1];
588                 }else if(getKomadai()[0][2] == 1) {
589                         board_i[5] = board_i[0];
590                 }else if(getKomadai()[1][2] == 1) {
591                         board_i[5] = board_i[1];
592                 }
593
594                 // 先手の金の持ち駒
595                 if(getKomadai()[0][1] == 2) {
596                         board_i[2] = board_i[0];
597                         board_i[3] = board_i[0];
598                 }
599
600                 // 先手の銀の持ち駒
601                 if(getKomadai()[0][2] == 2) {
602                         board_i[4] = board_i[0];
603                         board_i[5] = board_i[0];
604                 }
605                 // 後手の金の持ち駒
606                 if(getKomadai()[1][1] == 2) {
607                         board_i[2] = board_i[1];
608                         board_i[3] = board_i[1];
609                 }
610                 // 後手の銀の持ち駒
611                 if(getKomadai()[1][2] == 2) {
612                         board_i[4] = board_i[1];
613                         board_i[5] = board_i[1];
614                 }
615
616
617
618                 // 16 進数で ArrayList に保存
619                 String bo = "";
620                 for(int j = 0; j < 7; ++j) {
621                         if(board_i[j] == 10) {
622                                 bo += 'a';
623                         }else if(board_i[j] == 11) {
624                                 bo += 'b';
625                         }else if(board_i[j] == 12) {
626                                 bo += 'c';
627                         }else if(board_i[j] == 13) {
628                                 bo += 'd';
629                         }else if(board_i[j] == 14) {
630                                 bo += 'e';
631                         }else if(board_i[j] == 15) {
632                                 bo += 'f';
633                         }else {
634                                 bo += String.valueOf(board_i[j]);
635                         }
636                 }
637
638
639                 int i = Integer.parseInt(bo, 16);
640
641                 return kingin1 + i;
642
643
644
645         }
646
647
648
649
650
651
652
```

```
653
654             /**
655              * 詰みの判定
656              * 王の移動可能な手がなければ詰み
657              * @param int playerNum プレイヤー番号
658              * @return 詰みかどうか
659              */
660             public boolean isMate(int playerNum) {
661                     return (movableList.size() == 0);
662             }
663
664             public boolean isChecked(int playerNum) {
665                     int file, rank;
666                     if(playerNum == 0) {
667                             file = gyoku.file(); // 王の座標
668                             rank = gyoku.rank();
669                             //自分の駒視点で左から時計回り
670                             //E-金からの王手
671                             if (board [rank][file-1] == E_KIN) return true; // 左からの王手
672                             else if (board [rank-1][file-1] == E_KIN) return true; // 左上からの王手
673                             else if (board [rank-1][file] == E_KIN) return true; // 上からの王手
674                             else if (board [rank-1][file+1] == E_KIN) return true; // 右上からの王手
675                             else if (board [rank][file+1] == E_KIN) return true; // 右からの王手
676                             else if (board [rank+1][file] == E_KIN) return true; // 下からの王手
677                             // E-銀からの王手
678                             else if (board [rank-1][file-1] == E_GIN) return true; // 左上からの王手
679                             else if (board [rank-1][file] == E_GIN) return true; // 上からの王手
680                             else if (board [rank-1][file+1] == E_GIN) return true; // 右上からの王手
681                             else if (board [rank+1][file+1] == E_GIN) return true; // 右下からの王手
682                             else if (board [rank+1][file-1] == E_GIN) return true; // 左下からの王手
683                             else return false;
684                     }else {
685                             file = e_gyoku.file(); // E-王の座標
686                             rank = e_gyoku.rank();
687                             //自分の駒視点
688                             //金 からの王手
689                             if (board [rank][file-1] == KIN) return true; // 左からの王手
690                             else if (board [rank-1][file] == KIN) return true; // 上からの王手
691                             else if (board [rank][file+1] == KIN) return true; // 右からの王手
692                             else if (board [rank+1][file+1] == KIN) return true; // 右下からの王手
693                             else if (board [rank+1][file] == KIN) return true; // 下からの王手
694                             else if (board [rank+1][file-1] == KIN) return true; // 左下からの王手
695                             // 銀からの王手
696                             else if (board [rank-1][file-1] == GIN) return true; // 左上からの王手
697                             else if (board [rank-1][file] == GIN) return true; // 上からの王手
698                             else if (board [rank-1][file+1] == GIN) return true; // 右上からの王手
699                             else if (board [rank+1][file+1] == GIN) return true; // 右下からの王手
700                             else if (board [rank+1][file-1] == GIN) return true; // 左下からの王手
701
702
703                             else return false;
704                     }
705             }
706             /**
707              * 金銀複数の駒からの王手の判定
708              * @param playerNum
709              * @return
710              */
711             public boolean ismultyChecked(int playerNum) {
712                     int file, rank;
713                     if(playerNum == 0) {
714                             file = gyoku.file(); // 王の座標
715                             rank = gyoku.rank();
716                             //自分の駒視点で左から時計回り
717                             //E-金からの王手
718                             if (board [rank][file-1] == E_KIN && (board [rank-1][file-1] == E_GIN
719                             || board [rank-1][file] == E_GIN || board [rank-1][file+1] == E_GIN
720                             || board [rank+1][file+1] == E_GIN || board [rank+1][file-1] == E_GIN)) return true; // 左
    からの e-kin 全ての e-gin 王手
721                             else if (board [rank-1][file-1] == E_KIN && (board [rank-1][file] == E_GIN
722                             || board [rank-1][file+1] == E_GIN || board [rank+1][file+1] == E_GIN
723                             || board [rank+1][file-1] == E_GIN)) return true; // 左上からの全ての e-gin 王手
724                             else if (board [rank-1][file] == E_KIN &&(board [rank-1][file-1] == E_GIN
725                             || board [rank-1][file+1] == E_GIN || board [rank+1][file+1] == E_GIN
726                             || board [rank+1][file-1] == E_GIN)) return true; // 上からの全ての e-gin 王手
727                             else if (board [rank-1][file+1] == E_KIN && (board [rank-1][file-1] == E_GIN
728                             || board [rank-1][file] == E_GIN || board [rank+1][file+1] == E_GIN
729                             || board [rank+1][file-1] == E_GIN)) return true; // 右上からの王手
```

```
730                         else if (board [rank][file+1] == E_KIN && (board [rank-1][file-1] == E_GIN
731                         || board [rank-1][file] == E_GIN || board [rank-1][file+1] == E_GIN
732                         || board [rank+1][file+1] == E_GIN || board [rank+1][file-1] == E_GIN)) return true; // 右
    からの王手
733                         else if (board [rank+1][file] == E_KIN && (board [rank-1][file-1] == E_GIN
734                         || board [rank-1][file] == E_GIN || board [rank-1][file+1] == E_GIN
735                         || board [rank+1][file+1] == E_GIN || board [rank+1][file-1] == E_GIN)) return true; // 下
    からの王手
736                         // E-銀からの王手
737                         else if (board [rank-1][file-1] == E_GIN && (board [rank][file-1] == E_KIN
738                         || board [rank-1][file] == E_KIN || board [rank-1][file+1] == E_KIN
739                         || board [rank][file+1] == E_KIN || board [rank+1][file] == E_KIN)) return true; // 左
    上からの王手
740                         else if (board [rank-1][file] == E_GIN && (board [rank][file-1] == E_KIN
741                         || board [rank-1][file-1] == E_KIN || board [rank-1][file+1] == E_KIN
742                         || board [rank][file+1] == E_KIN || board [rank+1][file] == E_KIN)) return true; // 上
    からの王手
743                         else if (board [rank-1][file+1] == E_GIN && (board [rank][file-1] == E_KIN
744                         || board [rank-1][file-1] == E_KIN || board [rank-1][file] == E_KIN
745                         || board [rank][file+1] == E_KIN || board [rank+1][file] == E_KIN)) return true; // 右
    上からの王手
746                         else if (board [rank+1][file+1] == E_GIN && (board [rank-1][file+1] == E_KIN
747                         || board [rank][file-1] == E_KIN || board [rank-1][file-1] == E_KIN
748                         || board [rank-1][file] == E_KIN || board [rank][file+1] == E_KIN
749                         || board [rank+1][file] == E_KIN)) return true; // 右下からの王手
750                         else if (board [rank+1][file-1] == E_GIN && (board [rank-1][file+1] == E_KIN
751                         || board [rank][file-1] == E_KIN || board [rank-1][file-1] == E_KIN
752                         || board [rank-1][file] == E_KIN || board [rank][file+1] == E_KIN
753                         || board [rank+1][file] == E_KIN)) return true; // 左下からの王手
754                         else return false;
755                 }else {
756                         file = e_gyoku.file(); // E-王の座標
757                         rank = e_gyoku.rank();
758                         //自分の駒視点
759                         //金 からの王手
760                         if (board [rank][file-1] == KIN && (board [rank-1][file-1] == GIN
761                         || board [rank-1][file] == GIN || board [rank-1][file+1] == GIN
762                         || board [rank+1][file+1] == GIN || board [rank+1][file-1] == GIN)) return true; // 左
    からの王手
763                         else if (board [rank-1][file] == KIN && (board [rank-1][file-1] == GIN
764                         || board [rank-1][file+1] == GIN || board [rank+1][file+1] == GIN
765                         || board [rank+1][file-1] == GIN)) return true; // 上からの王手
766                         else if (board [rank][file+1] == KIN || (board [rank-1][file-1] == GIN
767                         || board [rank-1][file] == GIN || board [rank-1][file+1] == GIN
768                         || board [rank+1][file+1] == GIN || board [rank+1][file-1] == GIN)) return true; // 右
    からの王手
769                         else if (board [rank+1][file+1] == KIN && (board [rank-1][file-1] == GIN
770                         || board [rank-1][file] == GIN || board [rank-1][file+1] == GIN
771                         ||  board [rank+1][file-1] == GIN)) return true; // 右下からの王手
772                         else if (board [rank+1][file] == KIN && (board [rank-1][file-1] == GIN
773                         || board [rank-1][file] == GIN || board [rank-1][file+1] == GIN
774                         || board [rank+1][file+1] == GIN || board [rank+1][file-1] == GIN)) return true; // 下
    からの王手
775                         else if (board [rank+1][file-1] == KIN && (board [rank-1][file-1] == GIN
776                         || board [rank-1][file] == GIN || board [rank-1][file+1] == GIN
777                         || board [rank+1][file+1] == GIN )) return true; // 左下からの王手
778                         // 銀からの王手
779                         else if (board [rank-1][file-1] == GIN && (board [rank][file-1] == KIN
780                         || board [rank-1][file] == KIN || board [rank][file+1] == KIN
781                         || board [rank+1][file+1] == KIN || board [rank+1][file] == KIN
782                         || board [rank+1][file-1] == KIN)) return true; // 左上からの王手
783                         else if (board [rank-1][file] == GIN && (board [rank][file-1] == KIN
784                         || board [rank][file+1] == KIN || board [rank+1][file+1] == KIN || board [rank+1][file] == KIN
785                         || board [rank+1][file-1] == KIN)) return true; // 上からの王手
786                         else if (board [rank-1][file+1] == GIN && (board [rank][file-1] == KIN
787                         || board [rank-1][file] == KIN || board [rank][file+1] == KIN
788                         || board [rank+1][file+1] == KIN || board [rank+1][file] == KIN
789                         || board [rank+1][file-1] == KIN)) return true; // 右上からの王手
790                         else if (board [rank+1][file+1] == GIN && (board [rank][file-1] == KIN
791                         || board [rank-1][file] == KIN || board [rank][file+1] == KIN
792                         ||  board [rank+1][file] == KIN || board [rank+1][file-1] == KIN)) return true; // 右 下
    からの王手
793                         else if (board [rank+1][file-1] == GIN && (board [rank][file-1] == KIN
794                         || board [rank-1][file] == KIN || board [rank][file+1] == KIN
795                         || board [rank+1][file+1] == KIN || board [rank+1][file] == KIN )) return true; // 左 下
    からの王手
796
797
```

```
798                     else return false;
799             }
800     }
801
802     /**
803      * 候補手の作成
804      * movableList に移動可能な手を保存する
805      * @param int playerNum プレイヤー番号
806      */
807     public void createMovableList(int playerNum) {
808             movableList = new ArrayList<>();
809             if(playerNum == 0) { // 先手の場合
810                     if(gyoku != null) { // 盤面に王がある場合
811                             movableList.addAll (gyoku.movableList (board));  // 王が移動可能な手
812                     }
813                     if(kin != null) { // 盤面に金がある場合
814                             movableList.addAll (kin.movableList(board));  //金が移動可能な手
815                     }
816                     if(gin != null) { // 盤面に銀がある場合
817                             movableList.addAll (gin.movableList(board)); //銀が移動可能な手
818                     }
819
820
821                     if(isChecked(0)) { // 王に王手が掛かっている場合
822                             for(int i = 0; i < movableList.size();) {
823                                     Board nextBoard = nextBoard(movableList.get(i), playerNum);
824                                     if(nextBoard.isChecked(0)) { // 移動後に王手が掛かっている場合
825                                             movableList.remove(i); // 移動後で王手にかかっている手を削除する
826                                     }else {
827                                             ++i;
828                                     }
829                             }
830                     }
831             }else { // 後手の場合
832                     if(e_gyoku != null) { // 盤面に E-王がある場合
833                             movableList.addAll (e_gyoku.movableList (board));  // E-王が移動可能な手
834                     }
835                     if(e_kin != null) { // 盤面に E-金がある場合
836                             movableList.addAll (e_kin.movableList(board)); // E-金が移動可能な手
837                     }
838                     if(e_gin != null) { // 盤面に E-銀がある場合
839                             movableList.addAll (e_gin.movableList(board)); // E-銀が移動可能な手
840                     }
841
842                     if(isChecked(1)) { // E-王に王手が掛かっている場合
843                             for(int i = 0; i < movableList.size();) {
844                                     Board nextBoard = nextBoard(movableList.get(i), playerNum);
845                                     if(nextBoard.isChecked(1)) { // 移動後に王手が掛かっている場合
846                                             movableList.remove(i); // 移動後で王手にかかっている手を削除する
847                                     }else {
848                                             ++i;
849                                     }
850                             }
851                     }
852             }
853     }
854
855     public Board nextBoard(NextMove nextMove, int playerNum) {
856             Board nextBoard = new Board (board);
857             int movingType = nextMove.type(); // 移動する駒の種類
858             Piece movingPiece = null; // 移動する駒
859             // 移動する駒を設定する
860             switch (movingType) {
861             case GYOKU :
862                     movingPiece = nextBoard.gyoku;
863                     break;
864             case KIN :
865                     movingPiece = nextBoard.kin;
866                     break;
867             case GIN :
868                     movingPiece = nextBoard.gin;
869                     break;
870             case E_GYOKU :
871                     movingPiece = nextBoard.e_gyoku;
872                     break;
873             case E_KIN :
874                     movingPiece = nextBoard.e_kin;
```

```
875                     break;
876             case E_GIN :
877                     movingPiece = nextBoard.e_gin;
878                     break;
879             }
880             //int currentFile = movingPiece.file(); // 移動する駒の現在の X 座標
881             //int currentRank = movingPiece.rank(); // 移動する駒の現在の X 座標
882             int nextFile = nextMove.nextFile(); // 移動先の X 座標
883             int nextRank = nextMove.nextRank(); // 移動先の Y 座標
884             nextBoard.movePiece (movingPiece, movingType, nextFile, nextRank); // 駒を移動させる
885             return nextBoard;
886     }
887     /**
888      * 局面を回す用の nevtBoard
889      * @param nextMove
890      * @param playerNum
891      * @return
892      */
893     public Board nextBoard1(NextMove nextMove, int [][] komadai) {
894             Board nextBoard = new Board (board,komadai);
895             int movingType = nextMove.type(); // 移動する駒の種類
896             Piece movingPiece = null; // 移動する駒
897             // 移動する駒を設定する
898             switch (movingType) {
899             case GYOKU :
900                     movingPiece = nextBoard.gyoku;
901                     break;
902             case KIN :
903                     movingPiece = nextBoard.kin;
904                     break;
905             case GIN :
906                     movingPiece = nextBoard.gin;
907                     break;
908             case E_GYOKU :
909                     movingPiece = nextBoard.e_gyoku;
910                     break;
911             case E_KIN :
912                     movingPiece = nextBoard.e_kin;
913                     break;
914             case E_GIN :
915                     movingPiece = nextBoard.e_gin;
916                     break;
917             }
918             //int currentFile = movingPiece.file(); // 移動する駒の現在の X 座標
919             //int currentRank = movingPiece.rank(); // 移動する駒の現在の X 座標
920             int nextFile = nextMove.nextFile(); // 移動先の X 座標
921             int nextRank = nextMove.nextRank(); // 移動先の Y 座標
922             nextBoard.movePiece (movingPiece, movingType, nextFile, nextRank); // 駒を移動させる
923             return nextBoard;
924
925     }
926
927
928     /**
929      * 指定した位置に駒を移動させ、棋譜を返す
930      * @param Piece piece 移動させる駒
931      * @param int type 移動させる駒の種類
932      * @param int nextFile  移動先の X 座標
933      * @param int nextRank  移動先の Y 座標
934      * @param int playerNum プレイヤー番号
935      */
936     public void movePiece(Piece piece, int type, int nextFile, int nextRank) {
937
938             if (board[nextRank][nextFile] != EMPTY) { // 移動先に駒がある場合
939                     removePiece (nextFile, nextRank); // 移動先にある駒を取り除く
940             }
941             board[piece.rank()][piece.file()] = EMPTY; // 移動前のマスを空白に
942             piece.move (nextFile, nextRank); // 駒を移動
943             board[piece.rank()][piece.file()] = type; // 移動後のマスを指定した駒に
944
945
946     }
947
948     public void removePiece(int nextFile, int nextRank) {
949             switch (board [nextRank][nextFile]) {
950             case GYOKU : // 移動先に王
951                     gyoku = null; // 王を取り除く
```

```
                        break;
                case KIN : // 移動先に金
                        kin = null; // 金 を取り除く
                        komadai.setKomadai(1,1); // 後手 e-金の持ち駒を増やす
                        value = 1;
                        break;
                case GIN : // 移動先に銀
                        gin = null; // 銀 を取り除く
                        komadai.setKomadai(2,1); // 後手 e-銀の持ち駒を増やす
                        value = 2;
                        break;
                case E_GYOKU : // 移動先に E-王
                        e_gyoku = null; // E-王を取り除く
                        break;
                case E_KIN : // 移動先に E-金
                        e_kin = null; // E-金 を取り除く
                        komadai.setKomadai(1,0); // 先手金の持ち駒を増やす
                        value = 3;
                        break;
                case E_GIN : // 移動先に E=銀
                        e_gin= null; // E-銀を取り除く
                        komadai.setKomadai(2,0); // 先手銀の持ち駒を増やす
                        value = 4;
                        break;
                }
        }

//      /**
//       * 現在の盤面を配列に保存する
//       */
//      public void recordBoard() {
//              if (moves + lookAheadMoves < maxMove) {
//                      for (int r=0; r<4; ++r) {
//                              for (int f=0; f<4; ++f) {
//                                      boardRecord [moves + lookAheadMoves][r][f] = board[r+1][f+1];
//                              }
//                      }
//              }
//      }









//      /**
//       * 手数をリセットする
//       */
//      public void resetMoves() {
//              moves = 0;
//              lookAheadMoves = 0;
//      }

//      /**
//       * 先読み値の設定
//       */
//      public void setMaxDepth(int depth) {
//              this.maxDepth = depth;
//      }

//      /**
//       * 勝者の番号を返す
//       * 先手勝ち:1, 後手勝ち:-1, 引き分け:0
//       * @return  勝者の番号
//       */
//      public int winner() {
//              return winner;
//      }
}
```

以下に本研究で作成した Kyokumen クラスのソースコードを示す.

```
package syogi;

```

```
3  public class Kyokumen {
4
5          //  先手の駒  盤面手前
6          final static int GYOKU = 1; // 王
7          final static int KIN = 2; // 金
8          final static int GIN = 3; // 銀
9
10         // 後手の駒 盤面奥
11         final static int E_GYOKU = 4; // 王
12         final static int E_KIN = 5; // 金
13         final static int E_GIN = 6; // 銀
14
15         final static int EMPTY = 0; // 空白
16         final static int BORDER = Integer.MAX_VALUE; // 盤外
17         //盤面を表す  王王金金銀銀 を 16 進数の 6 桁で表す
18         //          金銀の組み合わせを末尾の数字で表す 初期局面 C3D2E1A
19         // 0 1 2 3
20         // 4 5 6 7
21         // 8 9 A B
22         // C D E F
23
24         // 未確定 0 先手勝ち 1 後手勝ち 2 引き分け 3 到達不能 4
25         static byte[] aaa = new byte [536870912]; // 盤面記録用
26
27
28
29         // 将棋盤面
30         public int[][] boardx= {
31                         {BORDER, BORDER, BORDER, BORDER, BORDER, BORDER},
32                         {BORDER, EMPTY, EMPTY, EMPTY, EMPTY, BORDER},
33                         {BORDER, EMPTY, EMPTY, EMPTY, EMPTY, BORDER},
34                         {BORDER, EMPTY, EMPTY, EMPTY, EMPTY, BORDER},
35                         {BORDER, EMPTY, EMPTY, EMPTY, EMPTY, BORDER},
36                         {BORDER, BORDER, BORDER, BORDER, BORDER, BORDER}
37         };
38
39
40         private int[] board_i = new int [7]; // 盤面の要素
41
42         private Board bo;
43         private int teban; // teban の数字 先手 0 後手 1
44         private int undecided,win,loss,draw,no; // 未確定, 勝ち, 負け, 引き分け, 到達不能の局面数
45
46         // 駒
47         Piece gyoku, kin, gin, e_gyoku, e_kin, e_gin;
48         Piece piece, kin1, kin2, gin1, gin2 = null;
49
50
51
52
53
54         /*
55          * コンストラクタ
56          *  駒の初期設定
57          */
58         public Kyokumen() {
59                 for(int i=0;i<board_i.length;i++) {
60                         board_i[i]=0;
61                 }
62
63                 bo = new Board(boardx);
64                 // 駒のインスタンス生成
65                 gyoku = new Piece(1);
66                 kin = new Piece(2);
67                 gin = new Piece(3);
68
69                 e_gyoku = new Piece(-1);
70                 e_kin = new Piece(-2);
71                 e_gin = new Piece(-3);
72
73         }
74
75
76
77
78
79
80
```

```
81
82          /**
83           * 金銀の組み合わせをセット
84           * @param board
85           */
86          public void setKinGin(int board) {
87
88                  if(268435456 <= board) {
89                          board -= 268435456;
90                  }
91
92                  String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする
93                  // 0000
94                  if(Character.getNumericValue(formatStr.charAt(6)) == 0) {
95                          kin1 = kin;
96                          kin2 = kin;
97                          gin1 = gin;
98                          gin2 = gin;
99                          // 0001
100                 }else if(Character.getNumericValue(formatStr.charAt(6)) == 1) {
101                         kin1 = kin;
102                         kin2 = kin;
103                         gin1 = gin;
104                         gin2 = e_gin;
105                         // 0010
106                 }else if(Character.getNumericValue(formatStr.charAt(6)) == 2) {
107                         kin1 = kin;
108                         kin2 = kin;
109                         gin1 = e_gin;
110                         gin2 = gin;
111                         // 0011
112                 }else if(Character.getNumericValue(formatStr.charAt(6)) == 3) {
113                         kin1 = e_kin;
114                         kin2 = e_kin;
115                         gin1 = gin;
116                         gin2 = gin;
117                         // 0100
118                 }else if(Character.getNumericValue(formatStr.charAt(6)) == 4) {
119                         kin1 = kin;
120                         kin2 = e_kin;
121                         gin1 = gin;
122                         gin2 = gin;
123                         // 0101
124                 }else if(Character.getNumericValue(formatStr.charAt(6)) == 5) {
125                         kin1 = kin;
126                         kin2 = e_kin;
127                         gin1 = gin;
128                         gin2 = e_gin;
129                         // 0110
130                 }else if(Character.getNumericValue(formatStr.charAt(6)) == 6) {
131                         kin1 = kin;
132                         kin2 = e_kin;
133                         gin1 = e_gin;
134                         gin2 = gin;
135                         // 0111
136                 }else if(Character.getNumericValue(formatStr.charAt(6)) == 7) {
137                         kin1 = kin;
138                         kin2 = e_kin;
139                         gin1 = e_gin;
140                         gin2 = e_gin;
141                         // 1000
142                 }else if(Character.getNumericValue(formatStr.charAt(6)) == 8) {
143                         kin1 = e_kin;
144                         kin2 = kin;
145                         gin1 = gin;
146                         gin2 = gin;
147                         // 1001
148                 }else if(Character.getNumericValue(formatStr.charAt(6)) == 9) {
149                         kin1 = e_kin;
150                         kin2 = kin;
151                         gin1 = gin;
152                         gin2 = e_gin;
153                         // 1010
154                 }else if(formatStr.charAt(6) == 'a') {
155                         kin1 = e_kin;
156                         kin2 = kin;
157                         gin1 = e_gin;
```

```
                        gin2 = gin;
                        // 1011
                }else if(formatStr.charAt(6) == 'b') {
                        kin1 = e_kin;
                        kin2 = kin;
                        gin1 = e_gin;
                        gin2 = e_gin;
                        // 1100
                }else if(formatStr.charAt(6) == 'c') {
                        kin1 = e_kin;
                        kin2 = e_kin;
                        gin1 = gin;
                        gin2 = gin;
                        // 1101
                }else if(formatStr.charAt(6) == 'd') {
                        kin1 = e_kin;
                        kin2 = e_kin;
                        gin1 = gin;
                        gin2 = e_gin;
                        // 1110
                }else if(formatStr.charAt(6) == 'e') {
                        kin1 = e_kin;
                        kin2 = e_kin;
                        gin1 = e_gin;
                        gin2 = gin;
                        // 1111
                }else if(formatStr.charAt(6) == 'f') {
                        kin1 = e_kin;
                        kin2 = e_kin;
                        gin1 = e_gin;
                        gin2 = e_gin;
                }
        }
        /**
         * 7 桁の 16 進数から board を生成
         * @param board
         * @return
         */
        public void madeBoard(int board) {

                // 後手番の盤面生成
                if(268435456 <= board) {
                        board -= 268435456;
                        teban =  1;
                }

                String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする


                bo.resetBoard(); // Board をすべて empty に
                bo.resetKomadai(); // 駒台の値をリセット

                // 語尾の 16 進数の値で金銀それぞれの先手駒後手駒の区別をする. 0 先手駒 1 後手駒 例 初期局面 0101

                setKinGin(board);
                // 金銀が自玉と同じ場所なら持ち駒にする

                // kin1 と kin2,gin1,gin2 が持ち駒
                if(allMotigomacheck(board)) {
                        for (int i =0; i<2;i++) {
                                switch(i) {
                                case 0:
                                        piece = gyoku;
                                        break;
                                case 1:
                                        piece = e_gyoku;
                                        break;
                                }
                                if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                        bo.setPiece(piece, 1, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                        bo.setPiece(piece, 1, 2);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                        bo.setPiece(piece, 1, 3);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                        bo.setPiece(piece, 1, 4);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
```

```
235                                     bo.setPiece(piece, 2, 1);
236                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
237                                     bo.setPiece(piece, 2, 2);
238                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
239                                     bo.setPiece(piece, 2, 3);
240                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
241                                     bo.setPiece(piece, 2, 4);
242                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
243                                     bo.setPiece(piece, 3, 1);
244                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
245                                     bo.setPiece(piece, 3, 2);
246                             }else if(formatStr.charAt(i) == 'a') {
247                                     bo.setPiece(piece, 3, 3);
248                             }else if(formatStr.charAt(i) == 'b') {
249                                     bo.setPiece(piece, 3, 4);
250                             }else if(formatStr.charAt(i) == 'c') {
251                                     bo.setPiece(piece, 4, 1);
252                             }else if(formatStr.charAt(i) == 'd') {
253                                     bo.setPiece(piece, 4, 2);
254                             }else if(formatStr.charAt(i) == 'e') {
255                                     bo.setPiece(piece, 4, 3);
256                             }else if(formatStr.charAt(i) == 'f') {
257                                     bo.setPiece(piece, 4, 4);
258                             }
259
260                     }
261                     // kin1 と kin2,gin1 が持ち駒
262             }else if(kin1kin2gin1Check(board)) {
263                     for (int i =0; i<2;i++) {
264                             switch(i) {
265                             case 0:
266                                     piece = gyoku;
267                                     break;
268                             case 1:
269                                     piece = e_gyoku;
270                                     break;
271                             }
272                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
273                                     bo.setPiece(piece, 1, 1);
274                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
275                                     bo.setPiece(piece, 1, 2);
276                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
277                                     bo.setPiece(piece, 1, 3);
278                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
279                                     bo.setPiece(piece, 1, 4);
280                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
281                                     bo.setPiece(piece, 2, 1);
282                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
283                                     bo.setPiece(piece, 2, 2);
284                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
285                                     bo.setPiece(piece, 2, 3);
286                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
287                                     bo.setPiece(piece, 2, 4);
288                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
289                                     bo.setPiece(piece, 3, 1);
290                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
291                                     bo.setPiece(piece, 3, 2);
292                             }else if(formatStr.charAt(i) == 'a') {
293                                     bo.setPiece(piece, 3, 3);
294                             }else if(formatStr.charAt(i) == 'b') {
295                                     bo.setPiece(piece, 3, 4);
296                             }else if(formatStr.charAt(i) == 'c') {
297                                     bo.setPiece(piece, 4, 1);
298                             }else if(formatStr.charAt(i) == 'd') {
299                                     bo.setPiece(piece, 4, 2);
300                             }else if(formatStr.charAt(i) == 'e') {
301                                     bo.setPiece(piece, 4, 3);
302                             }else if(formatStr.charAt(i) == 'f') {
303                                     bo.setPiece(piece, 4, 4);
304                             }
305                     }//gin2
306                     for (int i =5; i<6;i++) {
307                             switch(i) {
308                             case 5:
309                                     piece = gin2;
310                                     break;
```

```
311                             }
312                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
313                                     bo.setPiece(piece, 1, 1);
314                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
315                                     bo.setPiece(piece, 1, 2);
316                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
317                                     bo.setPiece(piece, 1, 3);
318                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
319                                     bo.setPiece(piece, 1, 4);
320                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
321                                     bo.setPiece(piece, 2, 1);
322                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
323                                     bo.setPiece(piece, 2, 2);
324                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
325                                     bo.setPiece(piece, 2, 3);
326                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
327                                     bo.setPiece(piece, 2, 4);
328                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
329                                     bo.setPiece(piece, 3, 1);
330                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
331                                     bo.setPiece(piece, 3, 2);
332                             }else if(formatStr.charAt(i) == 'a') {
333                                     bo.setPiece(piece, 3, 3);
334                             }else if(formatStr.charAt(i) == 'b') {
335                                     bo.setPiece(piece, 3, 4);
336                             }else if(formatStr.charAt(i) == 'c') {
337                                     bo.setPiece(piece, 4, 1);
338                             }else if(formatStr.charAt(i) == 'd') {
339                                     bo.setPiece(piece, 4, 2);
340                             }else if(formatStr.charAt(i) == 'e') {
341                                     bo.setPiece(piece, 4, 3);
342                             }else if(formatStr.charAt(i) == 'f') {
343                                     bo.setPiece(piece, 4, 4);
344                             }
345                     }
346                     // kin1 と gin1,gin2 が持ち駒
347             }else if(kin1gin1gin2Check(board)) {
348                     for (int i =0; i<2;i++) {
349                             switch(i) {
350                             case 0:
351                                     piece = gyoku;
352                                     break;
353                             case 1:
354                                     piece = e_gyoku;
355                                     break;
356                             }
357                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
358                                     bo.setPiece(piece, 1, 1);
359                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
360                                     bo.setPiece(piece, 1, 2);
361                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
362                                     bo.setPiece(piece, 1, 3);
363                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
364                                     bo.setPiece(piece, 1, 4);
365                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
366                                     bo.setPiece(piece, 2, 1);
367                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
368                                     bo.setPiece(piece, 2, 2);
369                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
370                                     bo.setPiece(piece, 2, 3);
371                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
372                                     bo.setPiece(piece, 2, 4);
373                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
374                                     bo.setPiece(piece, 3, 1);
375                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
376                                     bo.setPiece(piece, 3, 2);
377                             }else if(formatStr.charAt(i) == 'a') {
378                                     bo.setPiece(piece, 3, 3);
379                             }else if(formatStr.charAt(i) == 'b') {
380                                     bo.setPiece(piece, 3, 4);
381                             }else if(formatStr.charAt(i) == 'c') {
382                                     bo.setPiece(piece, 4, 1);
383                             }else if(formatStr.charAt(i) == 'd') {
384                                     bo.setPiece(piece, 4, 2);
385                             }else if(formatStr.charAt(i) == 'e') {
386                                     bo.setPiece(piece, 4, 3);
```

```
387                             }else if(formatStr.charAt(i) == 'f') {
388                                     bo.setPiece(piece, 4, 4);
389                             }
390                     }// kin2
391                     for (int i =3; i<4;i++) {
392                             switch(i) {
393                             case 3:
394                                     piece = kin2;
395                                     break;
396                             }
397                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
398                                     bo.setPiece(piece, 1, 1);
399                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
400                                     bo.setPiece(piece, 1, 2);
401                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
402                                     bo.setPiece(piece, 1, 3);
403                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
404                                     bo.setPiece(piece, 1, 4);
405                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
406                                     bo.setPiece(piece, 2, 1);
407                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
408                                     bo.setPiece(piece, 2, 2);
409                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
410                                     bo.setPiece(piece, 2, 3);
411                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
412                                     bo.setPiece(piece, 2, 4);
413                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
414                                     bo.setPiece(piece, 3, 1);
415                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
416                                     bo.setPiece(piece, 3, 2);
417                             }else if(formatStr.charAt(i) == 'a') {
418                                     bo.setPiece(piece, 3, 3);
419                             }else if(formatStr.charAt(i) == 'b') {
420                                     bo.setPiece(piece, 3, 4);
421                             }else if(formatStr.charAt(i) == 'c') {
422                                     bo.setPiece(piece, 4, 1);
423                             }else if(formatStr.charAt(i) == 'd') {
424                                     bo.setPiece(piece, 4, 2);
425                             }else if(formatStr.charAt(i) == 'e') {
426                                     bo.setPiece(piece, 4, 3);
427                             }else if(formatStr.charAt(i) == 'f') {
428                                     bo.setPiece(piece, 4, 4);
429                             }
430                     }
431                     // kin1 と kin2,gin2 が持ち駒
432             }else if(kin1kin2gin2Check(board)) {
433                     for (int i =0; i<2;i++) {
434                             switch(i) {
435                             case 0:
436                                     piece = gyoku;
437                                     break;
438                             case 1:
439                                     piece = e_gyoku;
440                                     break;
441                             }
442                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
443                                     bo.setPiece(piece, 1, 1);
444                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
445                                     bo.setPiece(piece, 1, 2);
446                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
447                                     bo.setPiece(piece, 1, 3);
448                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
449                                     bo.setPiece(piece, 1, 4);
450                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
451                                     bo.setPiece(piece, 2, 1);
452                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
453                                     bo.setPiece(piece, 2, 2);
454                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
455                                     bo.setPiece(piece, 2, 3);
456                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
457                                     bo.setPiece(piece, 2, 4);
458                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
459                                     bo.setPiece(piece, 3, 1);
460                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
461                                     bo.setPiece(piece, 3, 2);
462                             }else if(formatStr.charAt(i) == 'a') {
```

```
463                                     bo.setPiece(piece, 3, 3);
464                             }else if(formatStr.charAt(i) == 'b') {
465                                     bo.setPiece(piece, 3, 4);
466                             }else if(formatStr.charAt(i) == 'c') {
467                                     bo.setPiece(piece, 4, 1);
468                             }else if(formatStr.charAt(i) == 'd') {
469                                     bo.setPiece(piece, 4, 2);
470                             }else if(formatStr.charAt(i) == 'e') {
471                                     bo.setPiece(piece, 4, 3);
472                             }else if(formatStr.charAt(i) == 'f') {
473                                     bo.setPiece(piece, 4, 4);
474                             }
475                     }//gin1
476                     for (int i =4; i<5;i++) {
477                             switch(i) {
478                             case 4:
479                                     piece = gin1;
480                                     break;
481                             }
482                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
483                                     bo.setPiece(piece, 1, 1);
484                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
485                                     bo.setPiece(piece, 1, 2);
486                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
487                                     bo.setPiece(piece, 1, 3);
488                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
489                                     bo.setPiece(piece, 1, 4);
490                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
491                                     bo.setPiece(piece, 2, 1);
492                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
493                                     bo.setPiece(piece, 2, 2);
494                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
495                                     bo.setPiece(piece, 2, 3);
496                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
497                                     bo.setPiece(piece, 2, 4);
498                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
499                                     bo.setPiece(piece, 3, 1);
500                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
501                                     bo.setPiece(piece, 3, 2);
502                             }else if(formatStr.charAt(i) == 'a') {
503                                     bo.setPiece(piece, 3, 3);
504                             }else if(formatStr.charAt(i) == 'b') {
505                                     bo.setPiece(piece, 3, 4);
506                             }else if(formatStr.charAt(i) == 'c') {
507                                     bo.setPiece(piece, 4, 1);
508                             }else if(formatStr.charAt(i) == 'd') {
509                                     bo.setPiece(piece, 4, 2);
510                             }else if(formatStr.charAt(i) == 'e') {
511                                     bo.setPiece(piece, 4, 3);
512                             }else if(formatStr.charAt(i) == 'f') {
513                                     bo.setPiece(piece, 4, 4);
514                             }
515                     }
516                     // kin2 と gin1,gin2 が持ち駒
517             }else if(kin2gin1gin2Check(board)) {
518                     for (int i =0; i<3;i++) {
519                             switch(i) {
520                             case 0:
521                                     piece = gyoku;
522                                     break;
523                             case 1:
524                                     piece = e_gyoku;
525                                     break;
526                             case 2:
527                                     piece = kin1;
528                                     break;
529                             }
530                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
531                                     bo.setPiece(piece, 1, 1);
532                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
533                                     bo.setPiece(piece, 1, 2);
534                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
535                                     bo.setPiece(piece, 1, 3);
536                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
537                                     bo.setPiece(piece, 1, 4);
538                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
```

```
539                                     bo.setPiece(piece, 2, 1);
540                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
541                                     bo.setPiece(piece, 2, 2);
542                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
543                                     bo.setPiece(piece, 2, 3);
544                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
545                                     bo.setPiece(piece, 2, 4);
546                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
547                                     bo.setPiece(piece, 3, 1);
548                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
549                                     bo.setPiece(piece, 3, 2);
550                             }else if(formatStr.charAt(i) == 'a') {
551                                     bo.setPiece(piece, 3, 3);
552                             }else if(formatStr.charAt(i) == 'b') {
553                                     bo.setPiece(piece, 3, 4);
554                             }else if(formatStr.charAt(i) == 'c') {
555                                     bo.setPiece(piece, 4, 1);
556                             }else if(formatStr.charAt(i) == 'd') {
557                                     bo.setPiece(piece, 4, 2);
558                             }else if(formatStr.charAt(i) == 'e') {
559                                     bo.setPiece(piece, 4, 3);
560                             }else if(formatStr.charAt(i) == 'f') {
561                                     bo.setPiece(piece, 4, 4);
562                             }
563                     }
564                     // kin1 と kin2 が持ち駒
565             }else if(kin1kin2Check(board)) {
566                     for (int i =0; i<2;i++) {
567                             switch(i) {
568                             case 0:
569                                     piece = gyoku;
570                                     break;
571                             case 1:
572                                     piece = e_gyoku;
573                                     break;
574                             }
575                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
576                                     bo.setPiece(piece, 1, 1);
577                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
578                                     bo.setPiece(piece, 1, 2);
579                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
580                                     bo.setPiece(piece, 1, 3);
581                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
582                                     bo.setPiece(piece, 1, 4);
583                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
584                                     bo.setPiece(piece, 2, 1);
585                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
586                                     bo.setPiece(piece, 2, 2);
587                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
588                                     bo.setPiece(piece, 2, 3);
589                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
590                                     bo.setPiece(piece, 2, 4);
591                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
592                                     bo.setPiece(piece, 3, 1);
593                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
594                                     bo.setPiece(piece, 3, 2);
595                             }else if(formatStr.charAt(i) == 'a') {
596                                     bo.setPiece(piece, 3, 3);
597                             }else if(formatStr.charAt(i) == 'b') {
598                                     bo.setPiece(piece, 3, 4);
599                             }else if(formatStr.charAt(i) == 'c') {
600                                     bo.setPiece(piece, 4, 1);
601                             }else if(formatStr.charAt(i) == 'd') {
602                                     bo.setPiece(piece, 4, 2);
603                             }else if(formatStr.charAt(i) == 'e') {
604                                     bo.setPiece(piece, 4, 3);
605                             }else if(formatStr.charAt(i) == 'f') {
606                                     bo.setPiece(piece, 4, 4);
607                             }
608                     }//gin1 gin2
609                     for (int i =4; i<6;i++) {
610                             switch(i) {
611                             case 4:
612                                     piece = gin1;
613                                     break;
614                             case 5:
```

```
615                                     piece = gin2;
616                                     break;
617                             }
618                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
619                                     bo.setPiece(piece, 1, 1);
620                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
621                                     bo.setPiece(piece, 1, 2);
622                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
623                                     bo.setPiece(piece, 1, 3);
624                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
625                                     bo.setPiece(piece, 1, 4);
626                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
627                                     bo.setPiece(piece, 2, 1);
628                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
629                                     bo.setPiece(piece, 2, 2);
630                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
631                                     bo.setPiece(piece, 2, 3);
632                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
633                                     bo.setPiece(piece, 2, 4);
634                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
635                                     bo.setPiece(piece, 3, 1);
636                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
637                                     bo.setPiece(piece, 3, 2);
638                             }else if(formatStr.charAt(i) == 'a') {
639                                     bo.setPiece(piece, 3, 3);
640                             }else if(formatStr.charAt(i) == 'b') {
641                                     bo.setPiece(piece, 3, 4);
642                             }else if(formatStr.charAt(i) == 'c') {
643                                     bo.setPiece(piece, 4, 1);
644                             }else if(formatStr.charAt(i) == 'd') {
645                                     bo.setPiece(piece, 4, 2);
646                             }else if(formatStr.charAt(i) == 'e') {
647                                     bo.setPiece(piece, 4, 3);
648                             }else if(formatStr.charAt(i) == 'f') {
649                                     bo.setPiece(piece, 4, 4);
650                             }
651                     }
652                     // kin1 と gin1 が持ち駒
653             }else if(kin1gin1Check(board)) {
654                     for (int i =0; i<2;i++) {
655                             switch(i) {
656                             case 0:
657                                     piece = gyoku;
658                                     break;
659                             case 1:
660                                     piece = e_gyoku;
661                                     break;
662                             }
663                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
664                                     bo.setPiece(piece, 1, 1);
665                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
666                                     bo.setPiece(piece, 1, 2);
667                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
668                                     bo.setPiece(piece, 1, 3);
669                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
670                                     bo.setPiece(piece, 1, 4);
671                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
672                                     bo.setPiece(piece, 2, 1);
673                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
674                                     bo.setPiece(piece, 2, 2);
675                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
676                                     bo.setPiece(piece, 2, 3);
677                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
678                                     bo.setPiece(piece, 2, 4);
679                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
680                                     bo.setPiece(piece, 3, 1);
681                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
682                                     bo.setPiece(piece, 3, 2);
683                             }else if(formatStr.charAt(i) == 'a') {
684                                     bo.setPiece(piece, 3, 3);
685                             }else if(formatStr.charAt(i) == 'b') {
686                                     bo.setPiece(piece, 3, 4);
687                             }else if(formatStr.charAt(i) == 'c') {
688                                     bo.setPiece(piece, 4, 1);
689                             }else if(formatStr.charAt(i) == 'd') {
690                                     bo.setPiece(piece, 4, 2);
```

```
691                             }else if(formatStr.charAt(i) == 'e') {
692                                     bo.setPiece(piece, 4, 3);
693                             }else if(formatStr.charAt(i) == 'f') {
694                                     bo.setPiece(piece, 4, 4);
695                             }
696                     }//kin2
697                     for (int i =3; i<4;i++) {
698                             switch(i) {
699                             case 3:
700                                     piece = kin2;
701                                     break;
702                             }
703                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
704                                     bo.setPiece(piece, 1, 1);
705                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
706                                     bo.setPiece(piece, 1, 2);
707                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
708                                     bo.setPiece(piece, 1, 3);
709                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
710                                     bo.setPiece(piece, 1, 4);
711                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
712                                     bo.setPiece(piece, 2, 1);
713                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
714                                     bo.setPiece(piece, 2, 2);
715                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
716                                     bo.setPiece(piece, 2, 3);
717                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
718                                     bo.setPiece(piece, 2, 4);
719                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
720                                     bo.setPiece(piece, 3, 1);
721                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
722                                     bo.setPiece(piece, 3, 2);
723                             }else if(formatStr.charAt(i) == 'a') {
724                                     bo.setPiece(piece, 3, 3);
725                             }else if(formatStr.charAt(i) == 'b') {
726                                     bo.setPiece(piece, 3, 4);
727                             }else if(formatStr.charAt(i) == 'c') {
728                                     bo.setPiece(piece, 4, 1);
729                             }else if(formatStr.charAt(i) == 'd') {
730                                     bo.setPiece(piece, 4, 2);
731                             }else if(formatStr.charAt(i) == 'e') {
732                                     bo.setPiece(piece, 4, 3);
733                             }else if(formatStr.charAt(i) == 'f') {
734                                     bo.setPiece(piece, 4, 4);
735                             }
736                     }//gin2
737                     for (int i =5; i<6;i++) {
738                             switch(i) {
739                             case 5:
740                                     piece = kin2;
741                                     break;
742                             }
743                             if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
744                                     bo.setPiece(piece, 1, 1);
745                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
746                                     bo.setPiece(piece, 1, 2);
747                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
748                                     bo.setPiece(piece, 1, 3);
749                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
750                                     bo.setPiece(piece, 1, 4);
751                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
752                                     bo.setPiece(piece, 2, 1);
753                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
754                                     bo.setPiece(piece, 2, 2);
755                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
756                                     bo.setPiece(piece, 2, 3);
757                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
758                                     bo.setPiece(piece, 2, 4);
759                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
760                                     bo.setPiece(piece, 3, 1);
761                             }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
762                                     bo.setPiece(piece, 3, 2);
763                             }else if(formatStr.charAt(i) == 'a') {
764                                     bo.setPiece(piece, 3, 3);
765                             }else if(formatStr.charAt(i) == 'b') {
766                                     bo.setPiece(piece, 3, 4);
```

```
                        }else if(formatStr.charAt(i) == 'c') {
                                bo.setPiece(piece, 4, 1);
                        }else if(formatStr.charAt(i) == 'd') {
                                bo.setPiece(piece, 4, 2);
                        }else if(formatStr.charAt(i) == 'e') {
                                bo.setPiece(piece, 4, 3);
                        }else if(formatStr.charAt(i) == 'f') {
                                bo.setPiece(piece, 4, 4);
                        }
                }
                // kin1 と gin2 が持ち駒
        }else if(kin1gin2Check(board)) {
                for (int i =0; i<2;i++) {
                        switch(i) {
                        case 0:
                                piece = gyoku;
                                break;
                        case 1:
                                piece = e_gyoku;
                                break;
                        }
                        if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                bo.setPiece(piece, 1, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                bo.setPiece(piece, 1, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                bo.setPiece(piece, 1, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                bo.setPiece(piece, 1, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
                                bo.setPiece(piece, 2, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
                                bo.setPiece(piece, 2, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
                                bo.setPiece(piece, 2, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                bo.setPiece(piece, 2, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
                                bo.setPiece(piece, 3, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
                                bo.setPiece(piece, 3, 2);
                        }else if(formatStr.charAt(i) == 'a') {
                                bo.setPiece(piece, 3, 3);
                        }else if(formatStr.charAt(i) == 'b') {
                                bo.setPiece(piece, 3, 4);
                        }else if(formatStr.charAt(i) == 'c') {
                                bo.setPiece(piece, 4, 1);
                        }else if(formatStr.charAt(i) == 'd') {
                                bo.setPiece(piece, 4, 2);
                        }else if(formatStr.charAt(i) == 'e') {
                                bo.setPiece(piece, 4, 3);
                        }else if(formatStr.charAt(i) == 'f') {
                                bo.setPiece(piece, 4, 4);
                        }
                }//kin2 gin1
                for (int i =3; i<5;i++) {
                        switch(i) {
                        case 3:
                                piece = kin2;
                                break;
                        case 4:
                                piece = gin1;
                                break;
                        }
                        if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                bo.setPiece(piece, 1, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                bo.setPiece(piece, 1, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                bo.setPiece(piece, 1, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                bo.setPiece(piece, 1, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
                                bo.setPiece(piece, 2, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
                                bo.setPiece(piece, 2, 2);
```

```
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
                                bo.setPiece(piece, 2, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                bo.setPiece(piece, 2, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
                                bo.setPiece(piece, 3, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
                                bo.setPiece(piece, 3, 2);
                        }else if(formatStr.charAt(i) == 'a') {
                                bo.setPiece(piece, 3, 3);
                        }else if(formatStr.charAt(i) == 'b') {
                                bo.setPiece(piece, 3, 4);
                        }else if(formatStr.charAt(i) == 'c') {
                                bo.setPiece(piece, 4, 1);
                        }else if(formatStr.charAt(i) == 'd') {
                                bo.setPiece(piece, 4, 2);
                        }else if(formatStr.charAt(i) == 'e') {
                                bo.setPiece(piece, 4, 3);
                        }else if(formatStr.charAt(i) == 'f') {
                                bo.setPiece(piece, 4, 4);
                        }
                }
                // kin2,gin1 が持ち駒
        }else if(kin2gin1Check(board)) {
                for (int i =0; i<3;i++) {
                        switch(i) {
                        case 0:
                                piece = gyoku;
                                break;
                        case 1:
                                piece = e_gyoku;
                                break;
                        case 2:
                                piece = kin1;
                                break;
                        }
                        if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                bo.setPiece(piece, 1, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                bo.setPiece(piece, 1, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                bo.setPiece(piece, 1, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                bo.setPiece(piece, 1, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
                                bo.setPiece(piece, 2, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
                                bo.setPiece(piece, 2, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
                                bo.setPiece(piece, 2, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                bo.setPiece(piece, 2, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
                                bo.setPiece(piece, 3, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
                                bo.setPiece(piece, 3, 2);
                        }else if(formatStr.charAt(i) == 'a') {
                                bo.setPiece(piece, 3, 3);
                        }else if(formatStr.charAt(i) == 'b') {
                                bo.setPiece(piece, 3, 4);
                        }else if(formatStr.charAt(i) == 'c') {
                                bo.setPiece(piece, 4, 1);
                        }else if(formatStr.charAt(i) == 'd') {
                                bo.setPiece(piece, 4, 2);
                        }else if(formatStr.charAt(i) == 'e') {
                                bo.setPiece(piece, 4, 3);
                        }else if(formatStr.charAt(i) == 'f') {
                                bo.setPiece(piece, 4, 4);
                        }
                }//gin2
                for (int i =5; i<6;i++) {
                        switch(i) {
                        case 5:
                                piece = gin2;
                                break;
                        }
```

```
                        if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                bo.setPiece(piece, 1, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                bo.setPiece(piece, 1, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                bo.setPiece(piece, 1, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                bo.setPiece(piece, 1, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
                                bo.setPiece(piece, 2, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
                                bo.setPiece(piece, 2, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
                                bo.setPiece(piece, 2, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                bo.setPiece(piece, 2, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
                                bo.setPiece(piece, 3, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
                                bo.setPiece(piece, 3, 2);
                        }else if(formatStr.charAt(i) == 'a') {
                                bo.setPiece(piece, 3, 3);
                        }else if(formatStr.charAt(i) == 'b') {
                                bo.setPiece(piece, 3, 4);
                        }else if(formatStr.charAt(i) == 'c') {
                                bo.setPiece(piece, 4, 1);
                        }else if(formatStr.charAt(i) == 'd') {
                                bo.setPiece(piece, 4, 2);
                        }else if(formatStr.charAt(i) == 'e') {
                                bo.setPiece(piece, 4, 3);
                        }else if(formatStr.charAt(i) == 'f') {
                                bo.setPiece(piece, 4, 4);
                        }
                }
                // kin2,gin2 が持ち駒
        }else if(kin2gin2Check(board)) {
                for (int i =0; i<3;i++) {
                        switch(i) {
                        case 0:
                                piece = gyoku;
                                break;
                        case 1:
                                piece = e_gyoku;
                                break;
                        case 2:
                                piece = kin1;
                                break;
                        }
                        if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                bo.setPiece(piece, 1, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                bo.setPiece(piece, 1, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                bo.setPiece(piece, 1, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                bo.setPiece(piece, 1, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
                                bo.setPiece(piece, 2, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
                                bo.setPiece(piece, 2, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
                                bo.setPiece(piece, 2, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                bo.setPiece(piece, 2, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
                                bo.setPiece(piece, 3, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
                                bo.setPiece(piece, 3, 2);
                        }else if(formatStr.charAt(i) == 'a') {
                                bo.setPiece(piece, 3, 3);
                        }else if(formatStr.charAt(i) == 'b') {
                                bo.setPiece(piece, 3, 4);
                        }else if(formatStr.charAt(i) == 'c') {
                                bo.setPiece(piece, 4, 1);
                        }else if(formatStr.charAt(i) == 'd') {
                                bo.setPiece(piece, 4, 2);
```

```
995                             }else if(formatStr.charAt(i) == 'e') {
996                                     bo.setPiece(piece, 4, 3);
997                             }else if(formatStr.charAt(i) == 'f') {
998                                     bo.setPiece(piece, 4, 4);
999                             }
1000                    }//gin1
1001                    for (int i =4; i<5;i++) {
1002                            switch(i) {
1003                            case 4:
1004                                    piece = gin1;
1005                                    break;
1006                            }
1007                            if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
1008                                    bo.setPiece(piece, 1, 1);
1009                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
1010                                    bo.setPiece(piece, 1, 2);
1011                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
1012                                    bo.setPiece(piece, 1, 3);
1013                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
1014                                    bo.setPiece(piece, 1, 4);
1015                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
1016                                    bo.setPiece(piece, 2, 1);
1017                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
1018                                    bo.setPiece(piece, 2, 2);
1019                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
1020                                    bo.setPiece(piece, 2, 3);
1021                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
1022                                    bo.setPiece(piece, 2, 4);
1023                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
1024                                    bo.setPiece(piece, 3, 1);
1025                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
1026                                    bo.setPiece(piece, 3, 2);
1027                            }else if(formatStr.charAt(i) == 'a') {
1028                                    bo.setPiece(piece, 3, 3);
1029                            }else if(formatStr.charAt(i) == 'b') {
1030                                    bo.setPiece(piece, 3, 4);
1031                            }else if(formatStr.charAt(i) == 'c') {
1032                                    bo.setPiece(piece, 4, 1);
1033                            }else if(formatStr.charAt(i) == 'd') {
1034                                    bo.setPiece(piece, 4, 2);
1035                            }else if(formatStr.charAt(i) == 'e') {
1036                                    bo.setPiece(piece, 4, 3);
1037                            }else if(formatStr.charAt(i) == 'f') {
1038                                    bo.setPiece(piece, 4, 4);
1039                            }
1040                    }
1041                    // gin1,gin2 が持ち駒
1042            }else if(gin1gin2Check(board)) {
1043                    for (int i =0; i<4;i++) {
1044                            switch(i) {
1045                            case 0:
1046                                    piece = gyoku;
1047                                    break;
1048                            case 1:
1049                                    piece = e_gyoku;
1050                                    break;
1051                            case 2:
1052                                    piece = kin1;
1053                                    break;
1054                            case 3:
1055                                    piece = kin2;
1056                                    break;
1057                            }
1058                            if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
1059                                    bo.setPiece(piece, 1, 1);
1060                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
1061                                    bo.setPiece(piece, 1, 2);
1062                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
1063                                    bo.setPiece(piece, 1, 3);
1064                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
1065                                    bo.setPiece(piece, 1, 4);
1066                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
1067                                    bo.setPiece(piece, 2, 1);
1068                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
1069                                    bo.setPiece(piece, 2, 2);
1070                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
```

```
                                        bo.setPiece(piece, 2, 3);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                        bo.setPiece(piece, 2, 4);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
                                        bo.setPiece(piece, 3, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
                                        bo.setPiece(piece, 3, 2);
                                }else if(formatStr.charAt(i) == 'a') {
                                        bo.setPiece(piece, 3, 3);
                                }else if(formatStr.charAt(i) == 'b') {
                                        bo.setPiece(piece, 3, 4);
                                }else if(formatStr.charAt(i) == 'c') {
                                        bo.setPiece(piece, 4, 1);
                                }else if(formatStr.charAt(i) == 'd') {
                                        bo.setPiece(piece, 4, 2);
                                }else if(formatStr.charAt(i) == 'e') {
                                        bo.setPiece(piece, 4, 3);
                                }else if(formatStr.charAt(i) == 'f') {
                                        bo.setPiece(piece, 4, 4);
                                }
                        }
                        // kin1 が持ち駒
                }else if(formatStr.charAt(0) == formatStr.charAt(2) || formatStr.charAt(1) == formatStr.charAt(2)) {
                        for(int i=0;i<2;i++) {
                                if(formatStr.charAt(i) == formatStr.charAt(2)){
                                        bo.setKomadai(kin, i);
                                }
                        }
                        for (int i =0; i<2;i++) {
                                switch(i) {
                                case 0:
                                        piece = gyoku;
                                        break;
                                case 1:
                                        piece = e_gyoku;
                                        break;
                                }
                                if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                        bo.setPiece(piece, 1, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                        bo.setPiece(piece, 1, 2);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                        bo.setPiece(piece, 1, 3);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                        bo.setPiece(piece, 1, 4);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
                                        bo.setPiece(piece, 2, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
                                        bo.setPiece(piece, 2, 2);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
                                        bo.setPiece(piece, 2, 3);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                        bo.setPiece(piece, 2, 4);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
                                        bo.setPiece(piece, 3, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
                                        bo.setPiece(piece, 3, 2);
                                }else if(formatStr.charAt(i) == 'a') {
                                        bo.setPiece(piece, 3, 3);
                                }else if(formatStr.charAt(i) == 'b') {
                                        bo.setPiece(piece, 3, 4);
                                }else if(formatStr.charAt(i) == 'c') {
                                        bo.setPiece(piece, 4, 1);
                                }else if(formatStr.charAt(i) == 'd') {
                                        bo.setPiece(piece, 4, 2);
                                }else if(formatStr.charAt(i) == 'e') {
                                        bo.setPiece(piece, 4, 3);
                                }else if(formatStr.charAt(i) == 'f') {
                                        bo.setPiece(piece, 4, 4);
                                }
                        } //kin2 gin1 gin2
                        for (int i =3; i<6;i++) {
                                switch(i) {
                                case 3:
                                        piece = kin2;
                                        break;
                                case 4:
```

```
                                        piece = gin1;
                                        break;
                                case 5:
                                        piece = gin2;
                                        break;
                                }
                                if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                        bo.setPiece(piece, 1, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                        bo.setPiece(piece, 1, 2);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                        bo.setPiece(piece, 1, 3);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                        bo.setPiece(piece, 1, 4);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
                                        bo.setPiece(piece, 2, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
                                        bo.setPiece(piece, 2, 2);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
                                        bo.setPiece(piece, 2, 3);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                        bo.setPiece(piece, 2, 4);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
                                        bo.setPiece(piece, 3, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
                                        bo.setPiece(piece, 3, 2);
                                }else if(formatStr.charAt(i) == 'a') {
                                        bo.setPiece(piece, 3, 3);
                                }else if(formatStr.charAt(i) == 'b') {
                                        bo.setPiece(piece, 3, 4);
                                }else if(formatStr.charAt(i) == 'c') {
                                        bo.setPiece(piece, 4, 1);
                                }else if(formatStr.charAt(i) == 'd') {
                                        bo.setPiece(piece, 4, 2);
                                }else if(formatStr.charAt(i) == 'e') {
                                        bo.setPiece(piece, 4, 3);
                                }else if(formatStr.charAt(i) == 'f') {
                                        bo.setPiece(piece, 4, 4);
                                }
                        }
                        // kin2 が持ち駒
                }else if(formatStr.charAt(0) == formatStr.charAt(3) || formatStr.charAt(1) == formatStr.charAt(3)) {
                        for(int i=0;i<2;i++) {
                                if(formatStr.charAt(i) == formatStr.charAt(3)){
                                        bo.setKomadai(kin, i);
                                }
                        }
                        for (int i =0; i<3;i++) {
                                switch(i) {
                                case 0:
                                        piece = gyoku;
                                        break;
                                case 1:
                                        piece = e_gyoku;
                                        break;
                                case 2:
                                        piece = kin1;
                                        break;
                                }
                                if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                        bo.setPiece(piece, 1, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                        bo.setPiece(piece, 1, 2);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                        bo.setPiece(piece, 1, 3);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                        bo.setPiece(piece, 1, 4);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
                                        bo.setPiece(piece, 2, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
                                        bo.setPiece(piece, 2, 2);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
                                        bo.setPiece(piece, 2, 3);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                        bo.setPiece(piece, 2, 4);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
```

```
1224                                    bo.setPiece(piece, 3, 1);
1225                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
1226                                    bo.setPiece(piece, 3, 2);
1227                            }else if(formatStr.charAt(i) == 'a') {
1228                                    bo.setPiece(piece, 3, 3);
1229                            }else if(formatStr.charAt(i) == 'b') {
1230                                    bo.setPiece(piece, 3, 4);
1231                            }else if(formatStr.charAt(i) == 'c') {
1232                                    bo.setPiece(piece, 4, 1);
1233                            }else if(formatStr.charAt(i) == 'd') {
1234                                    bo.setPiece(piece, 4, 2);
1235                            }else if(formatStr.charAt(i) == 'e') {
1236                                    bo.setPiece(piece, 4, 3);
1237                            }else if(formatStr.charAt(i) == 'f') {
1238                                    bo.setPiece(piece, 4, 4);
1239                            }
1240                    }//gin1 gin2
1241                    for (int i =4; i<6;i++) {
1242                            switch(i) {
1243                            case 4:
1244                                    piece = gin1;
1245                                    break;
1246                            case 5:
1247                                    piece = gin2;
1248                                    break;
1249                            }
1250                            if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
1251                                    bo.setPiece(piece, 1, 1);
1252                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
1253                                    bo.setPiece(piece, 1, 2);
1254                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
1255                                    bo.setPiece(piece, 1, 3);
1256                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
1257                                    bo.setPiece(piece, 1, 4);
1258                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
1259                                    bo.setPiece(piece, 2, 1);
1260                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
1261                                    bo.setPiece(piece, 2, 2);
1262                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
1263                                    bo.setPiece(piece, 2, 3);
1264                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
1265                                    bo.setPiece(piece, 2, 4);
1266                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
1267                                    bo.setPiece(piece, 3, 1);
1268                            }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
1269                                    bo.setPiece(piece, 3, 2);
1270                            }else if(formatStr.charAt(i) == 'a') {
1271                                    bo.setPiece(piece, 3, 3);
1272                            }else if(formatStr.charAt(i) == 'b') {
1273                                    bo.setPiece(piece, 3, 4);
1274                            }else if(formatStr.charAt(i) == 'c') {
1275                                    bo.setPiece(piece, 4, 1);
1276                            }else if(formatStr.charAt(i) == 'd') {
1277                                    bo.setPiece(piece, 4, 2);
1278                            }else if(formatStr.charAt(i) == 'e') {
1279                                    bo.setPiece(piece, 4, 3);
1280                            }else if(formatStr.charAt(i) == 'f') {
1281                                    bo.setPiece(piece, 4, 4);
1282                            }
1283                    }
1284                    //gin1 が持ち駒
1285            }else if(formatStr.charAt(0) == formatStr.charAt(4) || formatStr.charAt(1) == formatStr.charAt(4)) {
1286                    for(int i=0;i<2;i++) {
1287                            if(formatStr.charAt(i) == formatStr.charAt(4)){
1288                                    bo.setKomadai(gin, i);
1289                            }
1290                    }
1291
1292                    for (int i =0; i<4;i++) {
1293                            switch(i) {
1294                            case 0:
1295                                    piece = gyoku;
1296                                    break;
1297                            case 1:
1298                                    piece = e_gyoku;
1299                                    break;
1300                            case 2:
```

```
                                piece = kin1;
                                break;
                        case 3:
                                piece = kin2;
                                break;
                        }
                        if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                bo.setPiece(piece, 1, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                bo.setPiece(piece, 1, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                bo.setPiece(piece, 1, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                bo.setPiece(piece, 1, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
                                bo.setPiece(piece, 2, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
                                bo.setPiece(piece, 2, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
                                bo.setPiece(piece, 2, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                bo.setPiece(piece, 2, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
                                bo.setPiece(piece, 3, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
                                bo.setPiece(piece, 3, 2);
                        }else if(formatStr.charAt(i) == 'a') {
                                bo.setPiece(piece, 3, 3);
                        }else if(formatStr.charAt(i) == 'b') {
                                bo.setPiece(piece, 3, 4);
                        }else if(formatStr.charAt(i) == 'c') {
                                bo.setPiece(piece, 4, 1);
                        }else if(formatStr.charAt(i) == 'd') {
                                bo.setPiece(piece, 4, 2);
                        }else if(formatStr.charAt(i) == 'e') {
                                bo.setPiece(piece, 4, 3);
                        }else if(formatStr.charAt(i) == 'f') {
                                bo.setPiece(piece, 4, 4);
                        }
                }//gin2
                for (int i =5; i<6;i++) {
                        switch(i) {
                        case 5:
                                piece = gin2;
                                break;
                        }
                        if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                bo.setPiece(piece, 1, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                bo.setPiece(piece, 1, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                bo.setPiece(piece, 1, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                bo.setPiece(piece, 1, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
                                bo.setPiece(piece, 2, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
                                bo.setPiece(piece, 2, 2);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
                                bo.setPiece(piece, 2, 3);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                bo.setPiece(piece, 2, 4);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
                                bo.setPiece(piece, 3, 1);
                        }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
                                bo.setPiece(piece, 3, 2);
                        }else if(formatStr.charAt(i) == 'a') {
                                bo.setPiece(piece, 3, 3);
                        }else if(formatStr.charAt(i) == 'b') {
                                bo.setPiece(piece, 3, 4);
                        }else if(formatStr.charAt(i) == 'c') {
                                bo.setPiece(piece, 4, 1);
                        }else if(formatStr.charAt(i) == 'd') {
                                bo.setPiece(piece, 4, 2);
                        }else if(formatStr.charAt(i) == 'e') {
                                bo.setPiece(piece, 4, 3);
```

```
1377                                    }else if(formatStr.charAt(i) == 'f') {
1378                                            bo.setPiece(piece, 4, 4);
1379                                    }
1380                            }
1381                            // gin2 が持ち駒
1382                    }else if(formatStr.charAt(0) == formatStr.charAt(5) || formatStr.charAt(1) == formatStr.charAt(5)) {
1383                            for(int i=0;i<2;i++) {
1384                                    if(formatStr.charAt(i) == formatStr.charAt(5)){
1385                                            bo.setKomadai(gin, i);
1386                                    }
1387                            }
1388                            for (int i =0; i<5;i++) {
1389                                    switch(i) {
1390                                    case 0:
1391                                            piece = gyoku;
1392                                            break;
1393                                    case 1:
1394                                            piece = e_gyoku;
1395                                            break;
1396                                    case 2:
1397                                            piece = kin1;
1398                                            break;
1399                                    case 3:
1400                                            piece = kin2;
1401                                            break;
1402                                    case 4:
1403                                            piece = gin1;
1404                                            break;
1405                                    }
1406                                    if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
1407                                            bo.setPiece(piece, 1, 1);
1408                                    }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
1409                                            bo.setPiece(piece, 1, 2);
1410                                    }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
1411                                            bo.setPiece(piece, 1, 3);
1412                                    }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
1413                                            bo.setPiece(piece, 1, 4);
1414                                    }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
1415                                            bo.setPiece(piece, 2, 1);
1416                                    }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
1417                                            bo.setPiece(piece, 2, 2);
1418                                    }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
1419                                            bo.setPiece(piece, 2, 3);
1420                                    }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
1421                                            bo.setPiece(piece, 2, 4);
1422                                    }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
1423                                            bo.setPiece(piece, 3, 1);
1424                                    }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
1425                                            bo.setPiece(piece, 3, 2);
1426                                    }else if(formatStr.charAt(i) == 'a') {
1427                                            bo.setPiece(piece, 3, 3);
1428                                    }else if(formatStr.charAt(i) == 'b') {
1429                                            bo.setPiece(piece, 3, 4);
1430                                    }else if(formatStr.charAt(i) == 'c') {
1431                                            bo.setPiece(piece, 4, 1);
1432                                    }else if(formatStr.charAt(i) == 'd') {
1433                                            bo.setPiece(piece, 4, 2);
1434                                    }else if(formatStr.charAt(i) == 'e') {
1435                                            bo.setPiece(piece, 4, 3);
1436                                    }else if(formatStr.charAt(i) == 'f') {
1437                                            bo.setPiece(piece, 4, 4);
1438                                    }
1439                            }
1440                            // 全ての駒が盤面にある
1441                    }else {
1442                            for (int i =0; i<6;i++) {
1443                                    switch(i) {
1444                                    case 0:
1445                                            piece = gyoku;
1446                                            break;
1447                                    case 1:
1448                                            piece = e_gyoku;
1449                                            break;
1450                                    case 2:
1451                                            piece = kin1;
1452                                            break;
1453                                    case 3:
```

```
                                        piece = kin2;
                                        break;
                                case 4:
                                        piece = gin1;
                                        break;
                                case 5:
                                        piece = gin2;
                                        break;
                                }
                                if(Character.getNumericValue(formatStr.charAt(i)) == 0) {
                                        bo.setPiece(piece, 1, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 1) {
                                        bo.setPiece(piece, 1, 2);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 2) {
                                        bo.setPiece(piece, 1, 3);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 3) {
                                        bo.setPiece(piece, 1, 4);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 4) {
                                        bo.setPiece(piece, 2, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 5) {
                                        bo.setPiece(piece, 2, 2);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 6) {
                                        bo.setPiece(piece, 2, 3);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 7) {
                                        bo.setPiece(piece, 2, 4);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 8) {
                                        bo.setPiece(piece, 3, 1);
                                }else if(Character.getNumericValue(formatStr.charAt(i)) == 9) {
                                        bo.setPiece(piece, 3, 2);
                                }else if(formatStr.charAt(i) == 'a') {
                                        bo.setPiece(piece, 3, 3);
                                }else if(formatStr.charAt(i) == 'b') {
                                        bo.setPiece(piece, 3, 4);
                                }else if(formatStr.charAt(i) == 'c') {
                                        bo.setPiece(piece, 4, 1);
                                }else if(formatStr.charAt(i) == 'd') {
                                        bo.setPiece(piece, 4, 2);
                                }else if(formatStr.charAt(i) == 'e') {
                                        bo.setPiece(piece, 4, 3);
                                }else if(formatStr.charAt(i) == 'f') {
                                        bo.setPiece(piece, 4, 4);
                                }
                        }



                }

                bo = new Board(bo.getBorad(), bo.getKomadai());
                //              bo.resetKomadai();

                //              bo = new Board(bo.getBorad(), komadai);
                //              return new Board(bo.getBorad(), bo.getKomadai());

        }

//      /**
//       *
//       * @return
//       */
//      public byte[] getBoard() {
//              return this.aaa;
//      }



        public void isChecked() {

                int counter=0;


                for(int i =0;i<aaa.length;i++) {
                        if(checkMadeBoard(i)) {
                                madeBoard(i);
                                if(!( (bo.isChecked(0) && bo.isChecked(1)) || bo.ismultyChecked(0) || bo.ismultyChecked(1) )) {
                                        bo.createMovableList(teban);
                                        if(bo.isMate(teban)) {
```

```
1531                                        if(teban == 0)aaa[i]=2;//後手勝ち
1532                                        else if(teban == 1)aaa[i]=1;//先手勝ち
1533                                }
1534
1535                        }else {
1536                                aaa[i]=4;
1537                        }
1538                        // 手番じゃない玉に王手がかかっている
1539                        if((teban == 0 && bo.isChecked(1)) || (teban == 1 && bo.isChecked(0))) {
1540                                if(teban == 0)aaa[i]=1;//先手勝ち
1541                                else if(teban == 1)aaa[i]=2;//後手勝ち
1542
1543                        }
1544                }else {
1545                        aaa[i]=4;
1546                }
1547        }
1548
1549        for(int i =0;i<aaa.length;i++) {
1550                if(aaa[i]==0)undecided++;
1551                if(aaa[i]==1)win++;
1552                if(aaa[i]==2)loss++;
1553                if(aaa[i]==4)no++;
1554        }
1555        System.out.println("未確定"+undecided+"先手勝ち"+win+"後手勝ち"+loss+"到達不能"+no);
1556
1557        int x=0;
1558        int hozon = 0;
1559        int t=0;
1560
1561        while(t<1) {
1562                // 未確定の次の局面をしらべて 勝ち局面があれば勝ち、すべての手が負け確定なら負け
1563                for(int i =0;i<aaa.length;i++) {
1564                        if(aaa[i]==0) {
1565                                int num =0;
1566                                int sente=0; // 先手勝ち局面
1567                                int gote=0; // 後手勝ち局面
1568                                madeBoard(i);
1569                                // 持ち駒に金があれば、盤面に指す
1570                                if(bo.getKomadai()[teban][1] != 0) {
1571                                        for(int k = 1; k <= 4; ++k) {
1572                                                for(int j = 1; j <= 4; ++j) {
1573                                                        if(bo.getBorad()[k][j] == EMPTY) {
1574                                                                if(teban == 0) {
1575                                                                        bo.setPiece(kin, k, j);
1576                                                                }else {
1577                                                                        bo.setPiece(e_kin, k, j);
1578                                                                }
1579
1580                                                                bo.deleateKomadai(kin, teban);
1581
1582                                                                Board nextBoard = new Board(bo.getBorad(),bo.getKomadai());
1583                                                                if(teban ==0) {
1584                                                                        x = 268435456 + nextBoard.madeIndex();
1585                                                                }else {
1586                                                                        x=nextBoard.madeIndex();
1587                                                                }
1588                                                                if(aaa[x] == 1) {
1589                                                                        sente++;
1590                                                                }else if(aaa[x] == 2) {
1591                                                                        gote++;
1592                                                                }
1593
1594                                                                if(teban == 0) {
1595                                                                        bo.deleatePiece(kin, k, j);
1596                                                                }else {
1597                                                                        bo.deleatePiece(e_kin, k, j);
1598                                                                }
1599
1600                                                                bo.setKomadai(kin, teban);
1601                                                                num++;
1602                                                        }
1603                                                }
1604                                        }
1605                                }
1606                                // 持ち駒に銀があれば、盤面に指す
1607                                if(bo.getKomadai()[teban][2] != 0) {
1608                                        for(int k = 1; k <= 4; ++k) {
```

```
                                        for(int j = 1; j <= 4; ++j) {
                                                if(bo.getBorad()[k][j] == EMPTY) {
                                                        if(teban == 0) {
                                                                bo.setPiece(gin, k, j);
                                                        }else {
                                                                bo.setPiece(e_gin, k, j);
                                                        }

                                                        bo.deleateKomadai(gin, teban);

                                                        Board nextBoard = new Board(bo.getBorad(),bo.getKomadai());
                                                        if(teban ==0) {
                                                                x = 268435456 + nextBoard.madeIndex();
                                                        }else {
                                                                x=nextBoard.madeIndex();
                                                        }
                                                        if(aaa[x] == 1) {
                                                                sente++;
                                                        }else if(aaa[x] == 2) {
                                                                gote++;
                                                        }

                                                        if(teban == 0) {
                                                                bo.deleatePiece(gin, k, j);
                                                        }else {
                                                                bo.deleatePiece(e_gin, k, j);
                                                        }

                                                        bo.setKomadai(gin, teban);
                                                        num++;
                                                }
                                        }
                                }
                        }
                        // 盤面の指せる駒を指す
                        bo.createMovableList(teban);
                        for(int n = 0; n < bo.getMovableList().size();n++) {
                                Board nextBoard = bo.nextBoard1(bo.getMovableList().get(n), bo.getKomadai());
                                if(nextBoard.gyoku != null && nextBoard.e_gyoku !=null) {
                                        // teban 0 なら 1 にするためにたす
                                        if(teban ==0) {
                                                x = 268435456 + nextBoard.madeIndex();
                                        }else {
                                                x = nextBoard.madeIndex();
                                        }
                                        if(aaa[x] == 1) {
                                                sente++;
                                        }else if(aaa[x] == 2) {
                                                gote++;
                                        }
                                        num++;
                                }
                                if(nextBoard.getValue() == 1) nextBoard.deleateKomadai(kin, 1);
                                else if(nextBoard.getValue() == 2) nextBoard.deleateKomadai(gin, 1);
                                else if(nextBoard.getValue() == 3) nextBoard.deleateKomadai(kin, 0);
                                else if(nextBoard.getValue() == 4) nextBoard.deleateKomadai(gin, 0);


                        }
                        // 手番の勝ち局面があれば勝ち、すべての手が負け確定なら負け
                        if(teban == 0) {
                                if(sente !=0 ) {
                                        aaa[i] = 1;
                                }else if(num == gote) {
                                        aaa[i] = 2;
                                }
                        }else if(teban == 1) {
                                if(gote !=0 ) {
                                        aaa[i] = 2;
                                }else if(num == sente) {
                                        aaa[i] = 1;
                                }
                        }
                }
        }
        undecided=0;
        win=0;
        loss=0;
```

```
1687                    no=0;
1688                    for(int i =0;i<aaa.length;i++) {
1689                            if(aaa[i]==0)undecided++;
1690                            if(aaa[i]==1)win++;
1691                            if(aaa[i]==2)loss++;
1692                            if(aaa[i]==4)no++;
1693                    }
1694                    // 未確定が減らなくなったら終了
1695                    if(hozon == undecided) {
1696                            t++;
1697                    }
1698                    hozon = undecided;
1699                    System.out.print(counter);
1700                    System.out.println("未確定"+undecided+"先手勝ち"+win+"後手勝ち"+loss+"到達不能"+no);
1701                    counter++;
1702            }
1703
1704            System.out.println("初期局面 1"+aaa[205336085]);
1705            System.out.println("初期局面 2"+aaa[204657130]);
1706            System.out.println("初期局面 3"+aaa[204660249]);
1707            System.out.println("初期局面 4"+aaa[205332966]);
1708
1709
1710
1711            //              int i = 154224997+268435456;
1712            //              int i = 70345796;
1713            //              madeBoard(i);
1714            //              System.out.println(bo.madeIndex());
1715            //              bo.showBoard();
1716            //              bo.showKomadai();
1717            //              aaa[137902128] = 1;
1718            //              System.out.println("評価" +aaa[i]);
1719            //              bo.createMovableList(teban);
1720            //              bo.showKomadai();
1721            //              int[][] komadai = bo.getKomadai();
1722            //              System.out.println(teban);
1723            //
1724            //              for(int n = 0; n < bo.getMovableList().size();n++) {
1725            //                      System.out.println(n);
1726            //                      Board nextBoard = bo.nextBoard1(bo.getMovableList().get(n), bo.getKomadai());
1727            //                      nextBoard.showBoard();
1728            //                      nextBoard.showKomadai();
1729            //                      System.out.println(nextBoard.madeIndex());
1730            //                      if(nextBoard.getValue() == 1) nextBoard.deleateKomadai(kin, 1);
1731            //                      else if(nextBoard.getValue() == 2) nextBoard.deleateKomadai(gin, 1);
1732            //                      else if(nextBoard.getValue() == 3) nextBoard.deleateKomadai(kin, 0);
1733            //                      else if(nextBoard.getValue() == 4) nextBoard.deleateKomadai(gin, 0);
1734            //
1735            //
1736            //              }
1737            //              if(bo.getKomadai()[teban][1] != 0) {
1738            //                      for(int k = 1; k <= 4; ++k) {
1739            //                              for(int j = 1; j <= 4; ++j) {
1740            //                                      if(bo.getBorad()[k][j] == EMPTY) {
1741            //                                              if(teban == 0) {
1742            //                                                      bo.setPiece(kin, k, j);
1743            //                                              }else {
1744            //                                                      bo.setPiece(e_kin, k, j);
1745            //                                              }
1746            //                                              bo.showBoard();
1747            //                                              bo.deleateKomadai(kin, teban);
1748            //                                              bo.showKomadai();
1749            //                                              Board nextBoard = new Board(bo.getBorad(),bo.getKomadai());
1750            //                                              if(aaa[nextBoard.madeIndex()] == 1) {
1751            //                                                      aaa[i] = 1;
1752            //                                              }else if(aaa[nextBoard.madeIndex()] == 2) {
1753            //                                                      aaa[i] = 2;
1754            //                                              }
1755            //                                              System.out.println(nextBoard.madeIndex());
1756            //                                              if(teban == 0) {
1757            //                                                      bo.deleatePiece(kin, k, j);
1758            //                                              }else {
1759            //                                                      bo.deleatePiece(e_kin, k, j);
1760            //                                              }
1761            //
1762            //                                              bo.setKomadai(kin, teban);
1763            //
```

```
1764 //                                      }
1765 //                              }
1766 //                      }
1767 //              }
1768 //              bo.showKomadai();
1769 //              if(bo.getKomadai()[teban][2] != 0) {
1770 //                      for(int k = 1; k <= 4; ++k) {
1771 //                              for(int j = 1; j <= 4; ++j) {
1772 //                                      if(bo.getBorad()[k][j] == EMPTY) {
1773 //                                              if(teban == 0) {
1774 //                                                      bo.setPiece(gin, k, j);
1775 //                                              }else {
1776 //                                                      bo.setPiece(e_gin, k, j);
1777 //                                              }
1778 //                                              bo.showBoard();
1779 //                                              bo.deleateKomadai(gin, teban);
1780 //                                              bo.showKomadai();
1781 //                                              Board nextBoard = new Board(bo.getBorad(),bo.getKomadai());
1782 //                                              if(aaa[nextBoard.madeIndex()] == 1) {
1783 //                                                      aaa[i] = 1;
1784 //                                              }else if(aaa[nextBoard.madeIndex()] == 2) {
1785 //                                                      aaa[i] = 2;
1786 //                                              }
1787 //                                              System.out.println(nextBoard.madeIndex());
1788 //                                              if(teban == 0) {
1789 //                                                      bo.deleatePiece(gin, k, j);
1790 //                                              }else {
1791 //                                                      bo.deleatePiece(e_gin, k, j);
1792 //                                              }
1793 //
1794 //                                              bo.setKomadai(gin, teban);
1795 //
1796 //                                      }
1797 //                              }
1798 //                      }
1799 //              }
1800 //
1801 //              bo.showKomadai();
1802 //              System.out.println("評価" +aaa[i]);
1803
1804 //              aaa[154744880] = 1;
1805 //              aaa[154742832] =2;
1806 //              204555974
1807 //              System.out.println("評価" +aaa[i]);
1808
1809
1810
1811 ////           bo.createMovableList(teban);
1812 //             bo.createMovableList(0);
1813 //             if(bo.isMate(0)) {
1814 //                     System.out.println("罪です");
1815 //             }
1816 ////           bo.showmovableListsize();
1817 ////           int n =bo.madeIndex();
1818 //             System.out.println(bo.madeIndex());
1819 //             madeBoard(bo.madeIndex());
1820 //             bo.showBoard();
1821 //             bo.showKomadai();
1822 //             System.out.println(bo.madeIndex());
1823 //             madeBoard(bo.madeIndex());
1824 //             bo.showBoard();
1825 //             bo.showKomadai();System.out.println(bo.madeIndex());
1826 //             madeBoard(bo.madeIndex());
1827 //             bo.showBoard();
1828 //             bo.showKomadai();
1829
1830 //             System.out.println(bo.madeIndex());
1831 //             madeBoard(bo.madeIndex());
1832 //             bo.showBoard();
1833 //             bo.showKomadai();System.out.println(bo.madeIndex());
1834 //             madeBoard(bo.madeIndex());
1835 //             bo.showBoard();
1836 //             bo.showKomadai();
1837 //             System.out.println(bo.madeIndex());
1838 //             madeBoard(bo.madeIndex());
1839 //             bo.showBoard();
1840 //             bo.showKomadai();
1841 //             System.out.println(bo.madeIndex());
```

```
1842                    //              madeBoard(bo.madeIndex());
1843                    //              bo.showBoard();
1844                    //              bo.showKomadai();
1845
1846
1847                    //              bo.showKomadai();
1848
1849
1850                    //              if(isChecked(1616741)) {
1851                    //                      counter++;
1852                    //              }
1853                    //              if(bo.isChecked(0) && bo.isChecked(1)) {
1854                    //                      counter++;
1855                    //              }
1856
1857
1858
1859
1860            }
1861
1862
1863            /**
1864             * Board を生成できるかどうかを返す
1865             * @param board
1866             * @return
1867             */
1868            public boolean checkMadeBoard(int board) {
1869                    boolean ret = true;
1870                    if(268435456 <= board) {
1871                            board -= 268435456;
1872
1873                    }
1874
1875                    String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする
1876                    // 王と e-王が同じ場所だと生成できない
1877                    if(formatStr.charAt(0) == formatStr.charAt(1)) {
1878                            ret = false;
1879
1880                    }
1881                    // kin1kin2gin1gin2 が同じマスにある ab 0000
1882                    if((formatStr.charAt(0) != formatStr.charAt(2) && formatStr.charAt(1) != formatStr.charAt(2)) &&
1883                                    (formatStr.charAt(2) == formatStr.charAt(3) &&
1884                                    formatStr.charAt(2) == formatStr.charAt(4) && formatStr.charAt(2) == formatStr.charAt(5))) {
1885                            ret = false;
1886                    }
1887                    // kin1kin2gin1 ab 000x
1888                    if((formatStr.charAt(0) != formatStr.charAt(2) && formatStr.charAt(1) != formatStr.charAt(2)) &&
1889                                    (formatStr.charAt(2) == formatStr.charAt(3) &&
1890                                    formatStr.charAt(2) == formatStr.charAt(4) )) {
1891                            ret = false;
1892                    }
1893                    // kin1kin2gin2 ab 00x0
1894                    if((formatStr.charAt(0) != formatStr.charAt(2) && formatStr.charAt(1) != formatStr.charAt(2)) &&
1895                                    (formatStr.charAt(2) == formatStr.charAt(3) &&
1896                                    formatStr.charAt(2) == formatStr.charAt(5) )) {
1897                            ret = false;
1898                    }
1899                    // kin1gin1gin2 ab 0x00
1900                    if((formatStr.charAt(0) != formatStr.charAt(2) && formatStr.charAt(1) != formatStr.charAt(2)) &&
1901                                    (formatStr.charAt(2) == formatStr.charAt(4) &&
1902                                    formatStr.charAt(2) == formatStr.charAt(5) )) {
1903                            ret = false;
1904                    }
1905                    // kin2gin1gin2 ab x000
1906                    if((formatStr.charAt(0) != formatStr.charAt(3) && formatStr.charAt(1) != formatStr.charAt(3)) &&
1907                                    (formatStr.charAt(3) == formatStr.charAt(4) &&
1908                                    formatStr.charAt(3) == formatStr.charAt(5) )) {
1909                            ret = false;
1910                    }
1911                    // kin1kin2 ab 00xx
1912                    if((formatStr.charAt(0) != formatStr.charAt(2) && formatStr.charAt(1) != formatStr.charAt(2)) &&
1913                                    (formatStr.charAt(2) == formatStr.charAt(3))) {
1914                            ret = false;
1915                    }
1916                    // kin1gin1 ab 0x0x
1917                    if((formatStr.charAt(0) != formatStr.charAt(2) && formatStr.charAt(1) != formatStr.charAt(2)) &&
1918                                    (formatStr.charAt(2) == formatStr.charAt(4))) {
1919                            ret = false;
```

```
1920                    }
1921                    // kin1gin2 ab 0xx0
1922                    if((formatStr.charAt(0) != formatStr.charAt(2) && formatStr.charAt(1) != formatStr.charAt(2)) &&
1923                                    (formatStr.charAt(2) == formatStr.charAt(5))) {
1924                            ret = false;
1925                    }
1926                    // kin2gin1 ab x00x
1927                    if((formatStr.charAt(0) != formatStr.charAt(3) && formatStr.charAt(1) != formatStr.charAt(3)) &&
1928                                    (formatStr.charAt(3) == formatStr.charAt(4))) {
1929                            ret = false;
1930                    }
1931                    // kin2gin2 ab x0x0
1932                    if((formatStr.charAt(0) != formatStr.charAt(3) && formatStr.charAt(1) != formatStr.charAt(3)) &&
1933                                    (formatStr.charAt(3) == formatStr.charAt(5))) {
1934                            ret = false;
1935                    }
1936                    // gin1gin2 ab xx00
1937                    if((formatStr.charAt(0) != formatStr.charAt(4) && formatStr.charAt(1) != formatStr.charAt(4)) &&
1938                                    (formatStr.charAt(4) == formatStr.charAt(5))) {
1939                            ret = false;
1940                    }
1941
1942
1943                    return ret;
1944            }
1945
1946            /**
1947             * 金銀全てが持ち駒の場合の判定
1948             * @return
1949             *
1950             */
1951            public boolean allMotigomacheck(int board) {
1952                    boolean ret = false;
1953                    if(268435456 <= board) {
1954                            board -= 268435456;
1955                    }
1956                    String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする
1957                    // kin1 と kin2,gin1,gin2 が持ち駒 1000000 or 0100000/ 10 1111 01 0000
1958                    if((formatStr.charAt(0) == formatStr.charAt(2) || formatStr.charAt(1) == formatStr.charAt(2)) &&
1959                                    formatStr.charAt(2) == formatStr.charAt(3) && formatStr.charAt(2) == formatStr.charAt(4)
1960                                    && formatStr.charAt(2) == formatStr.charAt(5)) {
1961                            ret = true;
1962                            for(int i=0;i<2;i++) {
1963                                    if(formatStr.charAt(i) == formatStr.charAt(2)){
1964                                            bo.setKomadai(kin, i);
1965                                            bo.setKomadai(kin, i);
1966                                            bo.setKomadai(gin, i);
1967                                            bo.setKomadai(gin, i);
1968                                    }
1969                            }
1970
1971
1972                    }
1973                    // 10 0001 / 01 1110
1974                    if(((formatStr.charAt(0) == formatStr.charAt(5)) && (formatStr.charAt(1) == formatStr.charAt(2)))
1975                                    &&      (formatStr.charAt(2) == formatStr.charAt(3) && formatStr.charAt(2) == formatStr.charAt(4))
1976                                    ) {
1977                            ret = true;
1978                            bo.setKomadai(kin, 1);
1979                            bo.setKomadai(kin, 1);
1980                            bo.setKomadai(gin, 1);
1981                            bo.setKomadai(gin, 0);
1982                    }
1983                    // 10 0010 / 01 1101
1984                    if(((formatStr.charAt(0) == formatStr.charAt(4)) && (formatStr.charAt(1) == formatStr.charAt(2)))
1985                                    &&      (formatStr.charAt(2) == formatStr.charAt(3) && formatStr.charAt(2) == formatStr.charAt(5))
1986                                    ) {
1987                            ret = true;
1988                            bo.setKomadai(kin, 1);
1989                            bo.setKomadai(kin, 1);
1990                            bo.setKomadai(gin, 0);
1991                            bo.setKomadai(gin, 1);
1992                    }
1993                    // 10 0011 / 01 1100
1994                    if(((formatStr.charAt(0) == formatStr.charAt(4)) && (formatStr.charAt(1) == formatStr.charAt(2)))
1995                                    &&      (formatStr.charAt(2) == formatStr.charAt(3) && formatStr.charAt(4) == formatStr.charAt(5))
1996                                    ) {
1997                            ret = true;
```

```
1998                    bo.setKomadai(kin, 1);
1999                    bo.setKomadai(kin, 1);
2000                    bo.setKomadai(gin, 0);
2001                    bo.setKomadai(gin, 0);
2002            }
2003            // 10 0100 / 01 1011
2004            if(((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(2)))
2005                            &&      (formatStr.charAt(2) == formatStr.charAt(4) && formatStr.charAt(2) == formatStr.charAt(5))
2006                            ) {
2007                    ret = true;
2008                    bo.setKomadai(kin, 1);
2009                    bo.setKomadai(kin, 0);
2010                    bo.setKomadai(gin, 1);
2011                    bo.setKomadai(gin, 1);
2012            }
2013            // 10 0101 / 01 1010
2014            if(((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(2)))
2015                            &&      (formatStr.charAt(2) == formatStr.charAt(4) && formatStr.charAt(3) == formatStr.charAt(5))
2016                            ) {
2017                    ret = true;
2018                    bo.setKomadai(kin, 1);
2019                    bo.setKomadai(kin, 0);
2020                    bo.setKomadai(gin, 1);
2021                    bo.setKomadai(gin, 0);
2022            }
2023            // 10 0110 / 01 1001
2024            if(((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(2)))
2025                            &&      (formatStr.charAt(3) == formatStr.charAt(4) && formatStr.charAt(2) == formatStr.charAt(5))
2026                            ) {
2027                    ret = true;
2028                    bo.setKomadai(kin, 1);
2029                    bo.setKomadai(kin, 0);
2030                    bo.setKomadai(gin, 0);
2031                    bo.setKomadai(gin, 1);
2032            }
2033            // 10 0111 / 01 1000
2034            if(((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(2)))
2035                            &&      (formatStr.charAt(3) == formatStr.charAt(4) && formatStr.charAt(3) == formatStr.charAt(5))
2036                            ) {
2037                    ret = true;
2038                    bo.setKomadai(kin, 1);
2039                    bo.setKomadai(kin, 0);
2040                    bo.setKomadai(gin, 0);
2041                    bo.setKomadai(gin, 0);
2042            }
2043            // 10 1000 / 01 0111
2044            if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(3)))
2045                            &&      (formatStr.charAt(3) == formatStr.charAt(4) && formatStr.charAt(3) == formatStr.charAt(5))
2046                            ) {
2047                    ret = true;
2048                    bo.setKomadai(kin, 0);
2049                    bo.setKomadai(kin, 1);
2050                    bo.setKomadai(gin, 1);
2051                    bo.setKomadai(gin, 1);
2052            }
2053            // 10 1001 / 01 0110
2054            if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(3)))
2055                            &&      (formatStr.charAt(3) == formatStr.charAt(4) && formatStr.charAt(2) == formatStr.charAt(5))
2056                            ) {
2057                    ret = true;
2058                    bo.setKomadai(kin, 0);
2059                    bo.setKomadai(kin, 1);
2060                    bo.setKomadai(gin, 1);
2061                    bo.setKomadai(gin, 0);
2062            }
2063            // 10 1010 / 01 0101
2064            if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(3)))
2065                            &&      (formatStr.charAt(3) == formatStr.charAt(5) && formatStr.charAt(2) == formatStr.charAt(4))
2066                            ) {
2067                    ret = true;
2068                    bo.setKomadai(kin, 0);
2069                    bo.setKomadai(kin, 1);
2070                    bo.setKomadai(gin, 0);
2071                    bo.setKomadai(gin, 1);
2072            }
2073            // 10 1011 / 01 0100
2074            if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(3)))
2075                            &&      (formatStr.charAt(2) == formatStr.charAt(4) && formatStr.charAt(2) == formatStr.charAt(5))
```

```
                        ) {
                ret = true;
                bo.setKomadai(kin, 0);
                bo.setKomadai(kin, 1);
                bo.setKomadai(gin, 0);
                bo.setKomadai(gin, 0);
        }
        // 10 1100 / 01 0011
        if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(4)))
                        &&      (formatStr.charAt(2) == formatStr.charAt(3) && formatStr.charAt(4) == formatStr.charAt(5))
                        ) {
                ret = true;
                bo.setKomadai(kin, 0);
                bo.setKomadai(kin, 0);
                bo.setKomadai(gin, 1);
                bo.setKomadai(gin, 1);
        }
        // 10 1101 / 01 0010
        if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(4)))
                        &&      (formatStr.charAt(2) == formatStr.charAt(3) && formatStr.charAt(2) == formatStr.charAt(5))
                        ) {
                ret = true;
                bo.setKomadai(kin, 0);
                bo.setKomadai(kin, 0);
                bo.setKomadai(gin, 1);
                bo.setKomadai(gin, 0);
        }
        // 10 1110 / 01 0001
        if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(5)))
                        &&      (formatStr.charAt(2) == formatStr.charAt(3) && formatStr.charAt(2) == formatStr.charAt(4))
                        ) {
                ret = true;
                bo.setKomadai(kin, 0);
                bo.setKomadai(kin, 0);
                bo.setKomadai(gin, 0);
                bo.setKomadai(gin, 1);
        }
        return ret;
}


/**
 * kin1kin2gin1 が持ち駒の場合の判定
 * @return
 */
public boolean kin1kin2gin1Check(int board) {
        boolean ret = false;
        if(268435456 <= board) {
                board -= 268435456;
        }
        String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする
        // 10 000x / 01 111x
        if((formatStr.charAt(0) == formatStr.charAt(2) || formatStr.charAt(1) == formatStr.charAt(2)) &&
                        formatStr.charAt(2) == formatStr.charAt(3) && formatStr.charAt(2) == formatStr.charAt(4)
                        ) {
                ret = true;
                for(int i=0;i<2;i++) {
                        if(formatStr.charAt(i) == formatStr.charAt(2)){
                                bo.setKomadai(kin, i);
                                bo.setKomadai(kin, i);
                                bo.setKomadai(gin, i);
                        }
                }
        }
        // 10 001x / 01 110x
        if(((formatStr.charAt(0) == formatStr.charAt(4)) && (formatStr.charAt(1) == formatStr.charAt(2)))
                        &&      formatStr.charAt(2) == formatStr.charAt(3)
                        ) {
                ret = true;
                bo.setKomadai(kin, 1);
                bo.setKomadai(kin, 1);
                bo.setKomadai(gin, 0);
        }
        // 10 010x / 01 101x
        if(((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(2)))
                        &&      formatStr.charAt(2) == formatStr.charAt(4)
                        ) {
                ret = true;
```

```
                        bo.setKomadai(kin, 1);
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(gin, 1);
                }
                // 10 011x / 01 100x
                if(((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(2)))
                                &&      formatStr.charAt(3) == formatStr.charAt(4)
                                ) {
                        ret = true;
                        bo.setKomadai(kin, 1);
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(gin, 0);
                }
                // 10 100x / 01 011x
                if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(3)))
                                &&      formatStr.charAt(3) == formatStr.charAt(4)
                                ) {
                        ret = true;
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(kin, 1);
                        bo.setKomadai(gin, 1);
                }
                // 10 101x / 01 010x
                if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(3)))
                                &&      formatStr.charAt(2) == formatStr.charAt(4)
                                ) {
                        ret = true;
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(kin, 1);
                        bo.setKomadai(gin, 0);
                }
                // 10 110x / 01 001x
                if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(4)))
                                &&      formatStr.charAt(2) == formatStr.charAt(3)
                                ) {
                        ret = true;
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(gin, 1);
                }
                return ret;
        }
        /**
         * kin1gin1gin2 が持ち駒の場合の判定
         * @return
         */
        public boolean kin1gin1gin2Check(int board) {
                boolean ret = false;
                if(268435456 <= board) {
                        board -= 268435456;
                }
                String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする

                // 10 0x00 / 01 1x11
                if((formatStr.charAt(0) == formatStr.charAt(2) || formatStr.charAt(1) == formatStr.charAt(2)) &&
                                formatStr.charAt(2) == formatStr.charAt(4) && formatStr.charAt(2) == formatStr.charAt(5)) {
                        ret = true;
                        for(int i=0;i<2;i++) {
                                if(formatStr.charAt(i) == formatStr.charAt(2)){
                                        bo.setKomadai(kin, i);
                                        bo.setKomadai(gin, i);
                                        bo.setKomadai(gin, i);
                                }
                        }
                }
                // 10 0x01 / 01 1x10
                if(((formatStr.charAt(0) == formatStr.charAt(5)) && (formatStr.charAt(1) == formatStr.charAt(2)))
                                &&      formatStr.charAt(2) == formatStr.charAt(4)
                                ) {
                        ret = true;
                        bo.setKomadai(kin, 1);
                        bo.setKomadai(gin, 1);
                        bo.setKomadai(gin, 0);

                }
                // 10 0x10 / 01 1x01
                if(((formatStr.charAt(0) == formatStr.charAt(4)) && (formatStr.charAt(1) == formatStr.charAt(2)))
                                &&      formatStr.charAt(2) == formatStr.charAt(5)
```

```
                                ) {
                        ret = true;
                        bo.setKomadai(kin, 1);
                        bo.setKomadai(gin, 0);
                        bo.setKomadai(gin, 1);
                }
                // 10 0x11 / 01 1x00
                if(((formatStr.charAt(0) == formatStr.charAt(4)) && (formatStr.charAt(1) == formatStr.charAt(2)))
                                &&      formatStr.charAt(4) == formatStr.charAt(5)
                                ) {
                        ret = true;
                        bo.setKomadai(kin, 1);
                        bo.setKomadai(gin, 0);
                        bo.setKomadai(gin, 0);
                }
                // 10 1x00 / 01 0x11
                if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(4)))
                                &&      formatStr.charAt(4) == formatStr.charAt(5)
                                ) {
                        ret = true;
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(gin, 1);
                        bo.setKomadai(gin, 1);
                }
                // 10 1x01 / 01 0x10
                if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(4)))
                                &&      formatStr.charAt(2) == formatStr.charAt(5)
                                ) {
                        ret = true;
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(gin, 1);
                        bo.setKomadai(gin, 0);
                }
                // 10 1x10 / 01 0x01
                if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(5)))
                                &&      formatStr.charAt(2) == formatStr.charAt(4)
                                ) {
                        ret = true;
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(gin, 0);
                        bo.setKomadai(gin, 1);
                }
                return ret;
        }
        /**
         * kin1kin2gin2 が持ち駒の場合の判定
         * @return
         */
        public boolean kin1kin2gin2Check(int board) {
                boolean ret = false;
                if(268435456 <= board) {
                        board -= 268435456;
                }
                String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする

                // 10 00x0 / 01 11x1
                if((formatStr.charAt(0) == formatStr.charAt(2) || formatStr.charAt(1) == formatStr.charAt(2)) &&
                                formatStr.charAt(2) == formatStr.charAt(3) && formatStr.charAt(2) == formatStr.charAt(5)) {
                        ret = true;
                        for(int i=0;i<2;i++) {
                                if(formatStr.charAt(i) == formatStr.charAt(2)){
                                        bo.setKomadai(kin, i);
                                        bo.setKomadai(kin, i);
                                        bo.setKomadai(gin, i);
                                }
                        }
                }
                // 10 00x1 / 01 11x0
                if(((formatStr.charAt(0) == formatStr.charAt(5)) && (formatStr.charAt(1) == formatStr.charAt(2)))
                                &&      formatStr.charAt(2) == formatStr.charAt(3)
                                ) {
                        ret = true;
                        bo.setKomadai(kin, 1);
                        bo.setKomadai(kin, 1);
                        bo.setKomadai(gin, 0);
                }
                // 10 01x0 / 01 10x1
                if(((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(2)))
```

```
2310                                    &&      formatStr.charAt(2) == formatStr.charAt(5)
2311                                    ) {
2312                            ret = true;
2313                            bo.setKomadai(kin, 1);
2314                            bo.setKomadai(kin, 0);
2315                            bo.setKomadai(gin, 1);
2316                    }
2317                    // 10 01x1 / 01 10x0
2318                    if(((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(2)))
2319                                    &&      formatStr.charAt(3) == formatStr.charAt(5)
2320                                    ) {
2321                            ret = true;
2322                            bo.setKomadai(kin, 1);
2323                            bo.setKomadai(kin, 0);
2324                            bo.setKomadai(gin, 0);
2325                    }
2326                    // 10 10x0 / 01 01x1
2327                    if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(3)))
2328                                    &&      formatStr.charAt(3) == formatStr.charAt(5)
2329                                    ) {
2330                            ret = true;
2331                            bo.setKomadai(kin, 0);
2332                            bo.setKomadai(kin, 1);
2333                            bo.setKomadai(gin, 1);
2334                    }
2335                    // 10 10x1 / 01 01x0
2336                    if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(3)))
2337                                    &&      formatStr.charAt(2) == formatStr.charAt(5)
2338                                    ) {
2339                            ret = true;
2340                            bo.setKomadai(kin, 0);
2341                            bo.setKomadai(kin, 1);
2342                            bo.setKomadai(gin, 0);
2343                    }
2344                    // 10 11x0 / 01 00x1
2345                    if(((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(5)))
2346                                    &&      formatStr.charAt(2) == formatStr.charAt(3)
2347                                    ) {
2348                            ret = true;
2349                            bo.setKomadai(kin, 0);
2350                            bo.setKomadai(kin, 0);
2351                            bo.setKomadai(gin, 1);
2352                    }
2353                    return ret;
2354            }
2355
2356            /**
2357             * kin2gin1gin2 が持ち駒の場合の判定
2358             * @return
2359             */
2360            public boolean kin2gin1gin2Check(int board) {
2361                    boolean ret = false;
2362                    if(268435456 <= board) {
2363                            board -= 268435456;
2364                    }
2365                    String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする
2366
2367                    // 10 x000 / 01 x111
2368                    if((formatStr.charAt(0) == formatStr.charAt(3) || formatStr.charAt(1) == formatStr.charAt(3)) &&
2369                                    formatStr.charAt(3) == formatStr.charAt(4) && formatStr.charAt(3) == formatStr.charAt(5)) {
2370                            ret = true;
2371                            for(int i=0;i<2;i++) {
2372                                    if(formatStr.charAt(i) == formatStr.charAt(3)){
2373                                            bo.setKomadai(kin, i);
2374                                            bo.setKomadai(gin, i);
2375                                            bo.setKomadai(gin, i);
2376                                    }
2377                            }
2378                    }
2379                    // 10 x001 / 01 x110
2380                    if(((formatStr.charAt(0) == formatStr.charAt(5)) && (formatStr.charAt(1) == formatStr.charAt(3)))
2381                                    &&      formatStr.charAt(3) == formatStr.charAt(4)
2382                                    ) {
2383                            ret = true;
2384                            bo.setKomadai(kin, 1);
2385                            bo.setKomadai(gin, 1);
2386                            bo.setKomadai(gin, 0);
2387                    }
```

```
2388                // 10 x010 / 01 x101
2389                if(((formatStr.charAt(0) == formatStr.charAt(4)) && (formatStr.charAt(1) == formatStr.charAt(3)))
2390                                &&      formatStr.charAt(3) == formatStr.charAt(5)
2391                                ) {
2392                        ret = true;
2393                        bo.setKomadai(kin, 1);
2394                        bo.setKomadai(gin, 0);
2395                        bo.setKomadai(gin, 1);
2396                }
2397                // 10 x011 / 01 x100
2398                if(((formatStr.charAt(0) == formatStr.charAt(4)) && (formatStr.charAt(1) == formatStr.charAt(3)))
2399                                &&      formatStr.charAt(4) == formatStr.charAt(5)
2400                                ) {
2401                        ret = true;
2402                        bo.setKomadai(kin, 1);
2403                        bo.setKomadai(gin, 0);
2404                        bo.setKomadai(gin, 0);
2405                }
2406                // 10 x100 / 01 x011
2407                if(((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(4)))
2408                                &&      formatStr.charAt(4) == formatStr.charAt(5)
2409                                ) {
2410                        ret = true;
2411                        bo.setKomadai(kin, 0);
2412                        bo.setKomadai(gin, 1);
2413                        bo.setKomadai(gin, 1);
2414                }
2415                // 10 x101 / 01 x010
2416                if(((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(4)))
2417                                &&      formatStr.charAt(3) == formatStr.charAt(5)
2418                                ) {
2419                        ret = true;
2420                        bo.setKomadai(kin, 0);
2421                        bo.setKomadai(gin, 1);
2422                        bo.setKomadai(gin, 0);
2423                }
2424                // 10 x110 / 01 x001
2425                if(((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(5)))
2426                                &&      formatStr.charAt(3) == formatStr.charAt(4)
2427                                ) {
2428                        ret = true;
2429                        bo.setKomadai(kin, 0);
2430                        bo.setKomadai(gin, 0);
2431                        bo.setKomadai(gin, 1);
2432                }
2433                return ret;
2434        }
2435
2436        /**
2437         * kin1kin2 が持ち駒の場合の判定
2438         * @return
2439         */
2440        public boolean kin1kin2Check(int board) {
2441                boolean ret = false;
2442                if(268435456 <= board) {
2443                        board -= 268435456;
2444                }
2445                String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする
2446
2447                // 10 00xx / 01 11xx
2448                if((formatStr.charAt(0) == formatStr.charAt(2) || formatStr.charAt(1) == formatStr.charAt(2)) &&
2449                                formatStr.charAt(2) == formatStr.charAt(3)) {
2450                        ret = true;
2451                        for(int i=0;i<2;i++) {
2452                                if(formatStr.charAt(i) == formatStr.charAt(2)){
2453                                        bo.setKomadai(kin, i);
2454                                        bo.setKomadai(kin, i);
2455                                }
2456                        }
2457                }
2458                // 10 01xx / 01 10xx
2459                if((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(2))) {
2460                        ret = true;
2461                        bo.setKomadai(kin, 1);
2462                        bo.setKomadai(kin, 0);
2463                }
2464                // 10 10xx / 01 01xx
2465                if((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(3))) {
```

```
                        ret = true;
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(kin, 1);
                }

                return ret;
        }

        /**
         * kin1gin1 が持ち駒の場合の判定
         * @return
         */
        public boolean kin1gin1Check(int board) {
                boolean ret = false;
                if(268435456 <= board) {
                        board -= 268435456;
                }
                String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする

                // 10 0x0x / 01 1x1x
                if((formatStr.charAt(0) == formatStr.charAt(2) || formatStr.charAt(1) == formatStr.charAt(2)) &&
                                formatStr.charAt(2) == formatStr.charAt(4)) {
                        ret = true;
                        for(int i=0;i<2;i++) {
                                if(formatStr.charAt(i) == formatStr.charAt(2)){
                                        bo.setKomadai(kin, i);
                                        bo.setKomadai(gin, i);
                                }
                        }
                }
                // 10 0x1x / 01 1x0x
                if((formatStr.charAt(0) == formatStr.charAt(4)) && (formatStr.charAt(1) == formatStr.charAt(2))) {
                        ret = true;
                        bo.setKomadai(kin, 1);
                        bo.setKomadai(gin, 0);
                }
                // 10 1x0x / 01 0x1x
                if((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(4))) {
                        ret = true;
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(gin, 1);
                }

                return ret;
        }
        /**
         * kin1gin2 が持ち駒の場合の判定
         * @return
         */
        public boolean kin1gin2Check(int board) {
                boolean ret = false;
                if(268435456 <= board) {
                        board -= 268435456;
                }
                String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする

                // 10 0xx0 / 01 1xx1
                if((formatStr.charAt(0) == formatStr.charAt(2) || formatStr.charAt(1) == formatStr.charAt(2)) &&
                                formatStr.charAt(2) == formatStr.charAt(5)) {
                        ret = true;
                        for(int i=0;i<2;i++) {
                                if(formatStr.charAt(i) == formatStr.charAt(2)){
                                        bo.setKomadai(kin, i);
                                        bo.setKomadai(gin, i);
                                }
                        }
                }
                // 10 0xx1 / 01 1xx0
                if((formatStr.charAt(0) == formatStr.charAt(5)) && (formatStr.charAt(1) == formatStr.charAt(2))) {
                        ret = true;
                        bo.setKomadai(kin, 1);
                        bo.setKomadai(gin, 0);
                }
                // 10 1xx0 / 01 0xx1
                if((formatStr.charAt(0) == formatStr.charAt(2)) && (formatStr.charAt(1) == formatStr.charAt(5))) {
                        ret = true;
                        bo.setKomadai(kin, 0);
                        bo.setKomadai(gin, 1);
```

```
2544                    }
2545
2546                    return ret;
2547            }
2548
2549            /**
2550             * kin2gin1 が持ち駒の場合の判定
2551             * @return
2552             */
2553            public boolean kin2gin1Check(int board) {
2554                    boolean ret = false;
2555                    if(268435456 <= board) {
2556                            board -= 268435456;
2557                    }
2558                    String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする
2559
2560                    // 10 x00x / 01 x11x
2561                    if((formatStr.charAt(0) == formatStr.charAt(3) || formatStr.charAt(1) == formatStr.charAt(3)) &&
2562                                    formatStr.charAt(3) == formatStr.charAt(4)) {
2563                            ret = true;
2564                            for(int i=0;i<2;i++) {
2565                                    if(formatStr.charAt(i) == formatStr.charAt(3)){
2566                                            bo.setKomadai(kin, i);
2567                                            bo.setKomadai(gin, i);
2568                                    }
2569                            }
2570                    }
2571                    // 10 x01x / 01 x10x
2572                    if((formatStr.charAt(0) == formatStr.charAt(4)) && (formatStr.charAt(1) == formatStr.charAt(3))) {
2573                            ret = true;
2574                            bo.setKomadai(kin, 1);
2575                            bo.setKomadai(gin, 0);
2576                    }
2577                    // 10 x10x / 01 x01x
2578                    if((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(4))) {
2579                            ret = true;
2580                            bo.setKomadai(kin, 0);
2581                            bo.setKomadai(gin, 1);
2582                    }
2583
2584                    return ret;
2585            }
2586
2587            /**
2588             * kin2gin2 が持ち駒の場合の判定
2589             * @return
2590             */
2591            public boolean kin2gin2Check(int board) {
2592                    boolean ret = false;
2593                    if(268435456 <= board) {
2594                            board -= 268435456;
2595                    }
2596                    String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする
2597
2598                    // 10 x0x0 / 01 x1x1
2599                    if((formatStr.charAt(0) == formatStr.charAt(3) || formatStr.charAt(1) == formatStr.charAt(3)) &&
2600                                    formatStr.charAt(3) == formatStr.charAt(5)) {
2601                            ret = true;
2602                            for(int i=0;i<2;i++) {
2603                                    if(formatStr.charAt(i) == formatStr.charAt(3)){
2604                                            bo.setKomadai(kin, i);
2605                                            bo.setKomadai(gin, i);
2606                                    }
2607                            }
2608                    }
2609                    // 10 x0x1 / 01 x1x0
2610                    if((formatStr.charAt(0) == formatStr.charAt(5)) && (formatStr.charAt(1) == formatStr.charAt(3))) {
2611                            ret = true;
2612                            bo.setKomadai(kin, 1);
2613                            bo.setKomadai(gin, 0);
2614                    }
2615                    // 10 x1x0 / 01 x0x1
2616                    if((formatStr.charAt(0) == formatStr.charAt(3)) && (formatStr.charAt(1) == formatStr.charAt(5))) {
2617                            ret = true;
2618                            bo.setKomadai(kin, 0);
2619                            bo.setKomadai(gin, 1);
2620                    }
2621
```

```
2622                return ret;
2623        }
2624
2625        /**
2626         * gin1gin2 が持ち駒の場合の判定
2627         * @return
2628         */
2629        public boolean gin1gin2Check(int board) {
2630                boolean ret = false;
2631                if(268435456 <= board) {
2632                        board -= 268435456;
2633                }
2634                String formatStr = String.format("%07x", board);  // 7 桁に成るまでを 0 埋めをする
2635
2636                // 10 xx00 / 01 xx11
2637                if((formatStr.charAt(0) == formatStr.charAt(4) || formatStr.charAt(1) == formatStr.charAt(4)) &&
2638                                formatStr.charAt(4) == formatStr.charAt(5)) {
2639                        ret = true;
2640                        for(int i=0;i<2;i++) {
2641                                if(formatStr.charAt(i) == formatStr.charAt(4)){
2642                                        bo.setKomadai(gin, i);
2643                                        bo.setKomadai(gin, i);
2644                                }
2645                        }
2646                }
2647                // 10 xx01 / 01 xx10
2648                if((formatStr.charAt(0) == formatStr.charAt(5)) && (formatStr.charAt(1) == formatStr.charAt(4))) {
2649                        ret = true;
2650                        bo.setKomadai(gin, 1);
2651                        bo.setKomadai(gin, 0);
2652                }
2653                // 10 xx10 / 01 xx01
2654                if((formatStr.charAt(0) == formatStr.charAt(4)) && (formatStr.charAt(1) == formatStr.charAt(5))) {
2655                        ret = true;
2656                        bo.setKomadai(gin, 0);
2657                        bo.setKomadai(gin, 1);
2658                }
2659
2660                return ret;
2661        }
2662
2663        /**
2664         * 現在の盤面を ArrayList に保存する
2665         */
2666        //              public void lastrecordBoard() {
2667        //                      for (int i=0; i<hashboardRecord.size(); ++i) {
2668        //                              if(!(isChecked(hashboardRecord.get(i)))) {
2669        //                                      lastboardRecord.add(hashboardRecord.get(i));
2670        //                              }
2671        //
2672        //                      }
2673        //                      showBoard(lastboardRecord.size());
2674        //
2675        //
2676        //              }
2677
2678  //    /**
2679  //     * 双方王手かどうか
2680  //     * @param board
2681  //     * @return
2682  //     */
2683  //    public boolean isChecked(int board) {
2684  //            madeBoard(board);
2685  //            return bo.isChecked(0) && bo.isChecked(1);
2686  //    }
2687
2688
2689
2690  //    public void showRecord() {
2691  //            //                      for (int i=0; i<boardRecord.size(); ++i) {
2692  //            //                              showBoard(boardRecord.get(i));
2693  //            //                      }
2694  //            System.out.println(aaa.length);
2695  //    }
2696
2697
2698  //    public void showBoard(long board_i) {
2699  //
```

```
2700 //              System.out.print(board_i);
2701 //
2702 //              System.out.println();
2703 //      }
2704
2705
2706 }
2707
2708
```