卒業研究報告書

題目

中将棋対戦ゲームの開発

指導教員

石水 隆　講師

報告者

10-1-037-0138

木ノ下 翔大

近畿大学理工学部情報学科

平成 27 年 1 月 31 日提出

概要

将棋・囲碁に代表されるボードゲームは二人零和完全情報ゲームに分類される。これらのゲームは、人工知能の研究対象として適しているため、現在盛んに研究されている。とりわけ、本将棋は競技人口が多く、非常によく研究されており、プロの将士に勝てるAIも登場している。将棋と類似したゲームに中将棋がある。中将棋は本将棋よりも大きな盤面でより多くの駒を使用するが、取った駒は取り捨てで本将棋のように持ち駒として打つことはできない。この中将棋も人工知能の対象として適していると考えられるが、一方、中将棋は競技人口も少なく、ほとんど研究されておらず、有望なAIも見受けられない。そこで本研究では、強い中将棋AIの作成を目指す。

本研究では、駒に価値を設定し盤面の評価値を求める、定跡データベースに従って指す、先読みにより詰みの手順を探す、といった本将棋の強いAIに使われている手法を中将棋に適用し、強いプログラムを作成する。

目次

1　序論

1.1 二人零和完全情報ゲーム

将棋や囲碁に代表されるボードゲームは二人零和完全情報ゲームに分類されており、世界中に様々なバリエーションが存在する[2]。

二人零和有限確定完全情報ゲームは双方最善手を打った場合、先手勝ち、後手勝ち、引き分けのどれになるかはゲーム開始時点で決定しており、理論上、全ての可能な局面を解析することができれば最善の手を打つことができる。しかし多くのボードゲームでは、可能な局面の総数が極めて大きいため、完全解析を行うことは不可能である。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超えている。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]。

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 6×6 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[9]。また、サイズ 4×4 の囲碁は双方最善手を打つと持碁(引き分け)[10]、5×5 の囲碁は黒の 25 目勝ちとなる[11]。

将棋については、盤面のサイズや使用する駒の種類を減らしたサイズ５五将棋[12]やゴロゴロ将棋[13]などのミニ将棋がある。5 五将棋はサイズ 5×5 の盤と 6 種類の駒、ゴロゴロ将棋はサイズ 5×6 の盤と 4 種類の駒を使用する。図 1、図 2 に 5 五将棋およびゴロゴロ将棋の盤と駒野初期配置を示す。これらは本将棋と比べて可能な局面数が少ない。しかしながら現在のところまだこれらは完全解析はされていない。

飛	角	銀	金	王
				歩
歩				
王	金	銀	角	飛

図 1 5 五将棋の駒の初期配置

銀	金	王	金	銀
	歩	歩	歩	
	歩	歩	歩	
銀	金	玉	金	銀

図 2 ゴロゴロ将棋の盤と駒の初期配置

キ	ラ	ゾ
	ひ	
	ひ	
ぞ	ラ	キ

図 3 動物将棋の盤面と初期配置

ド	パ	木
カ	ア	食

図 4 アンパンマン将棋の盤と駒の初期配置

完全解析されているミニ将棋として、どうぶつしょうぎ[14](以下動物将棋とする)やアンパンマンはじめてしょうぎ[15](以下アンパンマン将棋とする)がある。動物将棋はサイズ 3×4 の盤と、ライオン、象、キリン、ひよこの 4 種類の駒を使用し、アンパンマン将棋はサイズ 3×5 の盤とアンパンマン(ばいきんマン)、食パンマン(ホラーマン)、カレーパンマン(ドキンちゃん)の 3 種類の駒を使

用する幼児向けのミニ将棋である。図 3、4 に動物将棋およびアンパンマン将棋の盤と駒の初期配置を示す。動物将棋は完全解析により双方最善手を指した場合、78 手で後手が勝つことが判明している[16]。また、アンパンマン将棋は、双方最善手を指すと千日手で引き分けとなることが判明している[17]。

1.2 ゲーム AI の手法

可能な局面数が多いゲームに対して完全解析を行うことは困難である。そのようなゲームに対しては確実な最善手を得ることはできないが、局面の評価値計算、定跡データベース、一定手先の先読み、終盤での必勝読みと完全読み、モンテカルロ法[18]などを用いてより有利だと思われる手を選択することができる。

局面の評価値計算とは、局面を判断するための指標である評価値を導出することである。コンピュータ将棋の場合のパラメータは、一般的に、成り駒を含めた駒の価値や、相手の攻め駒と自分の玉との距離などの相対的な位置関係などを数値化したものが用いられている。AI の強さは評価関数の作り方に応じて決まるため、評価関数はできるだけ選挙区を適切に評価できるように工夫して作る必要がある。

定跡データベースとは、プロ将士などが指した実践譜をもとに編成したデータベースのことである。このデータベースを使用することでより強い AI になる。しかし、相手があえて定跡以外の手を指すなどして、データベースに無い局面が出てきたときにはこの手法は使えない。

一定手数の先読みとは、一定手数を先読みすることにより最善の手を打たせることである。たとえば、先読みの深さを 3 とし、局面の分岐数を x とした場合、3x の局面数を評価し、その中から最善の手を打つことができる。一般に先読みする手数が多いほど強い AI となるが、先読み手数の増加に伴い探索時間が指数的に増えるため、適度に枝切りをして探索範囲を減らす工夫をする必要がある。

必勝読みとは、オセロのように勝敗だけでなく石差も問題になるゲームの場合に、勝敗のみを読み切ることをいい、また石差までを読み切ることを完全読みという。必勝読みの方が計算時間が少なくて済むため、一般にまず必勝読みで価値を確定させ上で、残り手数が少なくなると完全読みに切り替えてより点数の高い勝ちを目指すことが多い。将棋では、終盤の読みや詰め将棋には完全読みが行われている。

モンテカルロ法とは、乱数を用いたシミュレーションを何度も行うことにより近似解を求める計算手法である。解析的に解くことができない問題でも、十分な回数のシミュレーションを行うことにより、近似的に解を求めることができる。問題によっては他の数値計算手法より簡単に適用できるが、高い精度を得ようとすると計算回数が膨大になってしまうという弱点もある。

以上の手法を用いることにより、完全解析を行わなくてもある程度の強さのプログラムを作ることが可能であり、ゲームによってはプロに勝つこともできる。

1.3 中将棋

中将棋[1]は、本将棋に類似したボードゲームである。中将棋は、12×12 の盤上で双方 21 種類 46 枚の駒を使用する。一方、中将棋では、取った駒は取り捨てであり、本将棋のように持ち駒として打つことはできない。

1.4 本研究の目的

本将棋は、非常に研究されていて、プロの将士に勝てる AI も登場している。一方、中将棋は、ほとんど研究されておらず、有望な AI も見受けられない。そこで、本研究では、同様に中将棋のより強いプログラムを作成することが目的である。

1.5 本報告書の構成

２節では、中将棋の駒の種類や動き、ルールについて記述する。３節では、現存の AI との対戦を踏まえての結論や今後の課題について記述する。

2　中将棋とは

2.1 特徴

中将棋には、本将棋と違った特徴がある。中将棋の主な特徴として以下のものがある。

・駒は取り捨てであり、本将棋のように持ち駒として打つことはできない

・12×12 の盤上で双方 21 種類 46 枚の駒を使用する

・駒が成るのは相手側から数えて 4 段目になる

・不成で敵陣に入ることもできるが、次の手では成ることができない（ただし、次の手で駒が取ることができる場合は例外。また、歩の場合は相手側の最前列まで行かないと成れない）

・太子という駒があり、玉が取られても負けにはならない（太子は初期配置にはなく、酔象という駒が成ると太子になる。

2.2 駒の種類と動き

2.2.1　駒の種類

中将棋には初期配置では 21 種類の駒があり、成り駒を合わせると 29 種類の駒がある。香車、銀将は本将棋とは違い、中将棋では白駒、堅行となる。また、金将は成ることが可能で飛車になる。ただし、本将棋で使用されている桂馬は中将棋では使用されない。図 5 に中将棋で使用される駒の一覧を示し、図 6 に中将棋の初期配置を示す。

駒名	成った駒名
歩兵	酔象（すいぞう）
仲人（ちゅうにん）	と金
猛豹（もうひょう）	角行
銅将	横行（おうぎょう）
銀将	堅行（しゅぎょう）
金将	飛車
盲虎（もうこ）	飛鹿（ひろく）
酔象	太子（たいし）
香車	白駒（はくく）
反車（へんしゃ）	鯨鯢（けいげい）
横行	奔猪（ほんちょ）
堅行	飛牛（ひぎゅう）
角行	龍馬（りゅうめ）
飛車	龍王
龍馬	角鷹（かくおう）
龍王	飛鷲（ひじゅう）
鳳凰（ほうおう）	奔王（ほんおう）
奔王	
麒麟（きりん）	獅子（しし）
獅子	
玉将	

図 5 中将棋で使用される駒の一覧

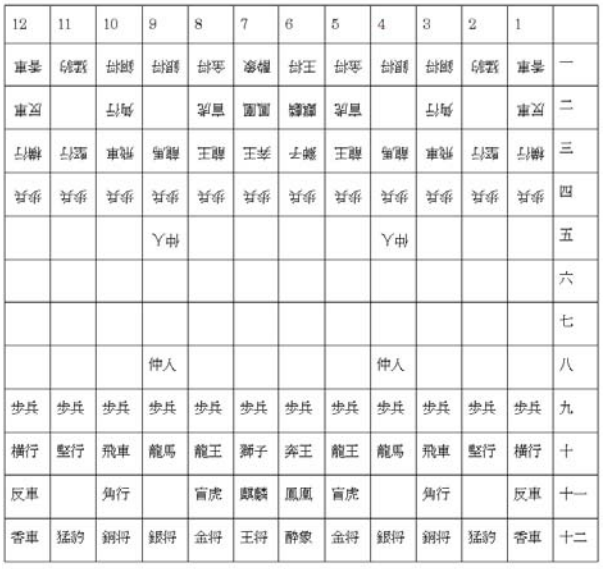

図 6 中将棋の初期配置[1]

2.2.2　駒の動き

中将棋の駒には、本将棋の龍王より強いチェスのクイーンと同じ動きをする奔王という駒が存在する。また、獅子、角鷹や飛鷲のような特殊な動きをする駒も存在する。本将棋にある駒を除いた中将棋の駒の動きの詳細を以下に示す。

・酔象：真後ろ以外の 1 目に進むことができる。成ると太子になる（図 7）

・太子：玉将と同じ動きをする。太子が盤面にある場合、玉将を取られても負にならない（図 8）

・銅将：前方 3 方向と後に 1 目進むことができる成ると横行になる（図 9）

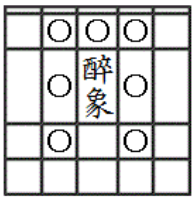

図 7 酔象の動き[19]

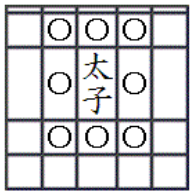

図 8 太子の動き[19]

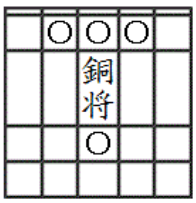

図 9 銅将の動き[19]

・横行：横方向に走ることができ、前後は１目進むことができる。成ると奔猪になる（図１０）

・奔猪：前後以外の六方向に走ることができる（図１１）

・猛豹：横方向以外に１目進むことができる。成ると角になる（図１２）

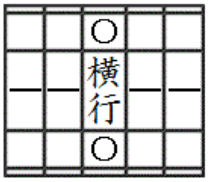

図 10 横行の動き[19]

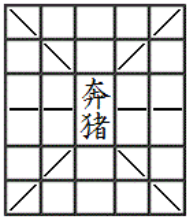

図 11 奔猪の動き[19]

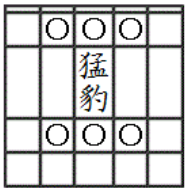

図 12 猛豹の動き[19]

・白駒：前３方向と後ろに走ることができる（図１３）

・麒麟：斜め方向に１目と前後左右に一目越して行くことができる。成ると獅子になる（図１４）

・奔王：８方向に走ることができる（図１５）

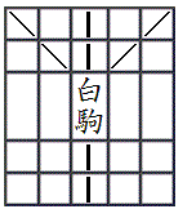

図 13 白駒の動き[19]

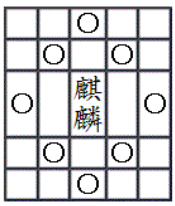

図 14 麒麟の動き[19]

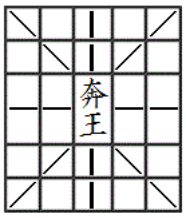

図 15 奔王の動き[19]

・盲虎：前以外に１目進むことができる。成ると飛鹿になる（図１６）

・飛鹿：前後に走ることができ、それ以外は１目進むことができる（図１７）

・反車：前後に走ることができる。成ると鯨鯢になる（図１８）

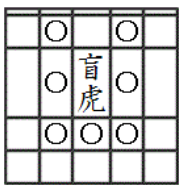

図 16 盲虎の動き[19]

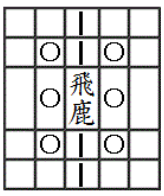

図 17 飛鹿の動き[19]

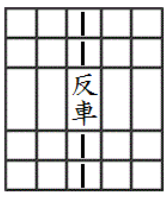

図 18 反車の動き[19]

・鯨鯢：後３方向と前に走ることができる（図１９）

・堅行：前後に走ることができ、横方向に１目進むことができる。成ると飛牛になる（図２０）

・飛牛：横方向以外に走ることができる（図２１）

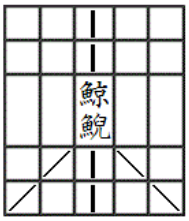

図 19 鯨鯢の動き[19]

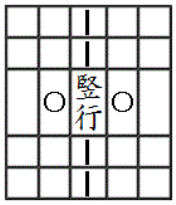

図 20 竪行の動き[19]

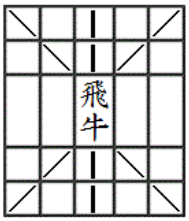

図 21 飛牛の動き[19]

・仲人：前後に１目進むことができ、成ると酔象になる（図２２）

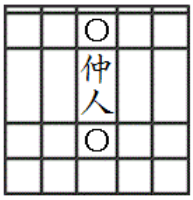

図 22 仲人の動き[19]

・獅子：⑴へ行く、⑴を飛び越え⑵に行く、⑴→⑵と２枚取りができる、⑴→その場に戻るという手のいずれかを指せる（図２３）

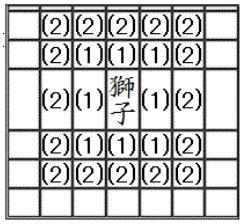

図 23 獅子の動き[19]

・鳳凰：縦横に１目進むことができ、斜め方向に１目越して行くことができる。成ると奔王になる（図２４）

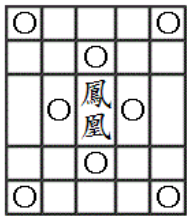

図 24 鳳凰の動き[19]

・飛鷲：後、横および後斜め方向に走ることができる。前斜め方向には獅子と同様に、⑴へ行く、⑴を飛び越え⑵に行く、⑴→⑵と２枚取りができる、⑴→その場に戻るという手のいずれかを指せる（図２５）

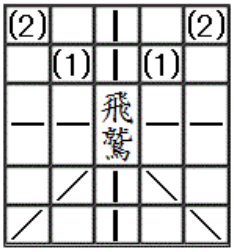

図 25 飛鷲の動き[19]

・角鷹前以外に走ることができる。前へは獅子と同様の動きができる（図２６）

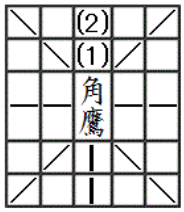

図 26 角鷹の動き[19]

2.3 ルール

本節では、中将棋特有のルールについて述べる

2.3.1　太子に関するルール

太子は初期配置では無く、酔象という駒が成ることで太子となる。太子は特殊な駒で玉将と同様の扱いとなる。そのため、玉将が相手側に取られた場合でも、太子が盤面にある限りゲームは終了せず、玉将と太子の両方を取ることでゲームが終了する。

2.3.2　じっと・居食い

「じっと」とは、2 マス移動可能な駒が現在の場所から 1 マス移動し、すぐに戻るということでその場から動かないという手法である。また、隣接するマスに敵の駒があれば、その駒を取り、元の場所に戻る「居食い」という手法もある。これらの手法を使用できる駒として、獅子、飛鷲や角鷹がある。

2.3.3　獅子に関する特殊ルール

獅子という駒には特殊なルールが存在する。これは、主に獅子が獅子を取り合うようなケースにおいて作られたものと考えられる。以下に獅子の特殊なルールを示す。

・獅子の足：獅子同士が 1 マス間を空けて隣り合っている場合(すなわち、お互い獅子の利きに入っている場合)で、獅子に足がある場合(足とはつなぎ駒のことで自分の獅子を守っている駒のこと)は、その獅子を取ることができない。図 5 の例では、どちらの獅子にも角行・飛車がつなぎ駒として味方の獅子を守っているので、獅子で獅子を取ることができない(つなぎ駒はどの駒での構わない)。ただし、獅子の足のルールが適用されるのは、獅子の間に 1 マス間が空いている場合で、獅子同士が隣接している場合は適用されず、無条件で取ることができる

・ウラ足、かげ足：自分の獅子に足が無く、足のある相手の獅子に責められた場合、その相手の獅子をまたぐ形で自分の獅子に走り駒の利きを置いた場合、ウラ足といい取ることができない。図 6 では、角行が敵側の獅子を挟んで獅子の足となっているので、敵側の獅子は取ることができなくなる

・先獅子：敵の獅子に自分の獅子以外の駒が当たった場合、無条件で取ることができる。この場合に、自分の獅子に足がある場合は先獅子というルールが適用され、敵側は直後の一手で獅子を取り返すとこができなくなる。図 7 では、どちらの獅子にも相手の駒が当たっている。この場合、先に相手の獅子を取った側は、自分の獅子に足が利いているのを条件に先獅子のルールが適用され、相手側は直後の一手で獅子を取り返すことができなくなる

・付け喰い：獅子と獅子の間に敵の駒がある場合、この駒を取った後、さらに、相手の獅子も一緒に取ることができる。ただし、歩兵と仲人は付け喰いの対象にはならない。また、付け喰いの後は、直後に敵のつなぎ駒で取り返すことが許されている。図 7 では、獅子と獅子の間に銀将という駒がある。この場合、銀将と獅子の 2 枚取りができる。この時の銀将を付け喰い駒といい、敵側が飛車で取り返した場合、獅子を討つと言う

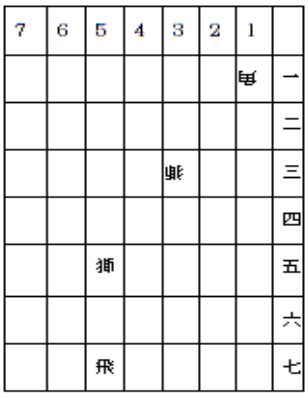

図 28 獅子の足の例[1]

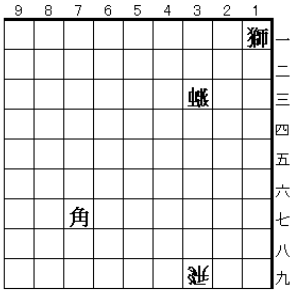

図 29 ウラ足・かげ足の例[1]

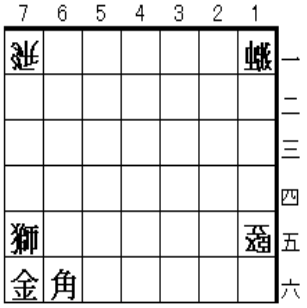

図 30 先獅子の例[1]

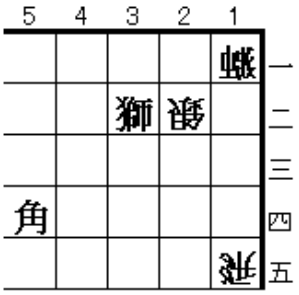

図 31 付け喰いの例[1]

3 本研究で作成したプログラム

本研究では Java 将棋アルゴリズム[6]を使用して将棋プログラムを作成し、それを基に中将棋のプログラムの作成を目指した。しかし、本プログラムでは対 CPU 戦をできるまでには至っておらず、本将棋のルールを用いた対人戦までしか完成しなかった。

本プログラムを作成するにあたっての問題に中将棋の特有のルールを実装することが挙げられる。中将棋の特有のルールには駒は取り捨て、太子ルール、獅子のルール、居食い・じっとがある。まず、駒の取り捨てについては既存のクラスから持ち駒に関する部分を変更する必要がある。

居食い・じっとや獅子のルールに関しては、駒の動きを設定する KomaMoves クラスで駒ごとに駒の動きを設定する必要がある。KomaMoves クラスでは、各方向への駒の動きを boolean 型の canMove メソッドや canJump メソッドを使用し、動ける場合は「true」、動けない場合は「false」で表している。同様に、駒ごとに動きを設定するメソッドを実装する必要がある。

また、太子に関しては、中将棋では玉将と同じ扱いになるので、合法手を生成する GenerateMoves クラスで王手のチェック等の玉将に関するルールの場所に追加する必要がある。

4 結論・今後の課題

本研究では、駒に価値を設定し盤面の評価値を求める、定跡データベースに従って指す、先読みにより詰みの手順を探す、といった本将棋の強い AI に使われている手法を中将棋に適用し、強いプログラムの作成を目指した。しかし、対 CPU 戦の実装までには至ることができなかった。

今後の課題としては、以下のことが挙げられる。現時点では本将棋のルールになっているので、太子ルールや駒を取り捨てにする等の中将棋特有のルールを実装する。また、上記のような方法を使用し、強いプログラムにする必要がある。

謝辞

本研究を作成するにあたり、指導教員の石水隆講師から、丁寧かつ熱心なご指導を賜りました。ここに感謝の意を表します。

参考文献

[1] 日本中将棋連盟：中将棋講座 http://www.chushogi-renmei.com/index.htm
[2] 松田道弘：世界のゲーム辞典、東京堂出版(1989)
[3] FM-SOFT,FM　Excel 中将棋、(2001) http://www.vector.co.jp/soft/win95/game/se214637.html
[4] SDIN 中将棋、SDIN 無料ゲーム、(2014) http:/sdin.jp/browser/board/chughogi
[5] 池 秦弘：コンピュータ将棋のアルゴリズム-最強アルゴリズムの探求とプログラミング、工学社(2005)
[6] 池 秦弘：Java 将棋のアルゴリズム、工学社(2007)
[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] 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
[10] 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), https://ipsj.ixsq.nii.ac.jp/ej/?action=repository_uri&item_id=58632
[11] 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)
[12] 日本 5 五将棋連盟、 http://www.geocities.co.jp/Playtown-Spade/8662/
[13] 「ごろごろどうぶつしょうぎ」発売開始！、 お知らせ、 日本将棋連盟、2012 年 11 月 26 日、(2012) http://www.shogi.or.jp/topics/2012/11/post-652.html
[14] 北尾まどか、藤田麻衣子、 どうぶつしょうぎねっと、(2010) http://dobutsushogi.net/
[15] アンパンマンはじめて将棋、セガトイズ (2012) http://www.segatoys.co.jp/anpan/product/popup/_legacy/learn/06.htm
[16] 田中哲郎：「どうぶつしょうぎ」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/
[17] 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、2013 年度 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/
[18] 美添一樹、 山下宏、 松原仁、 コンピュータ囲碁―モンテカルロ法の理論と実践―、 共立出版、(2012)
[19] 中将棋の駒の動かし方-Ne　http://www.ne.jp/asahi/tetsu/toybox/chushogi/koma.html

付録

Constants.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
//各種定数の定義
public interface Constants {
	//「先手」の定義
	public final static int SENTE = 1 << 6;
	//「後手」の定義
	public final static int GOTE = 1 << 7;
	//筋を表す文字列の定義
	public final static String sujiStr[] =
	{"","１","２","３","４","５","６","７","８","９","ａ","ｂ","ｃ"};
	//段を表す文字列の定義
	public final static String danStr[] =
	{"","一","二","三","四","五","六","七","八","九","Ａ","Ｂ","Ｃ"};//縦
}
```

Koma.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
public class Koma implements Constants, Cloneable {
	//駒の種類の定義
	public static final int EMPTY = 0;		//「空」
	public static final int EMP = EMPTY;		//「空」の別名
	public static final int PROMOTE = 21;		//「成り」フラグ

	public static final int FU = 1;		//「歩」
	public static final int KY = 2;		//「香」
	public static final int GI = 3;		//「銀」
	public static final int KI = 4;		//「金」
	public static final int KA = 5;		//「角」
	public static final int HI = 6;		//「飛」
	public static final int CH = 7;		//「仲」
	public static final int MO = 8;		//「猛」
	public static final int DO = 9;		//「銅」
	public static final int KO = 10;		//「虎」
	public static final int SU = 11;		//「酔」
	public static final int HE = 12;		//「反」
	public static final int YO = 13;		//「横」
	public static final int SH = 14;		//「堅」
	public static final int RY = 15;		//「馬」
```

```
public static final int RO = 16;                    //「龍」
public static final int HO = 17;                    //「鳳」
public static final int HN = 18;                    //「奔」
public static final int KN = 19;                    //「麒」
public static final int SS = 20;                    //「獅」
public static final int OU = 21;                    //「王」
public static final int TO = FU + PROMOTE; //「と金」=「歩」+ 成り
public static final int HA = KY + PROMOTE; //「白」
public static final int NG = GI + PROMOTE; //「成り銀」=「堅」
public static final int NI = KI + PROMOTE;  //「成り金」=「飛」
public static final int UM = KA + PROMOTE;//「成り角」=「馬」
public static final int NH = HI + PROMOTE; //「成り飛」=「龍」
public static final int NC = SU + PROMOTE; //「成り仲」=「酔」
public static final int NM = MO + PROMOTE;//「成り猛」=「角」
public static final int ND = DO + PROMOTE;//「成り銅」=「横」
public static final int NK = KO + PROMOTE;//「成り虎」=「鹿」
public static final int TA = SU + PROMOTE;  //「太」
public static final int KE = HE + PROMOTE; //「鯨」
public static final int NY = YO + PROMOTE; //「成り横」=「猪」
public static final int NS = SH + PROMOTE; //「牛」
public static final int NO = RY + PROMOTE; //「鷹」
public static final int HJ = RO + PROMOTE; //「鷲」
public static final int NN = HN + PROMOTE;//「成り鳳」=「奔」
public static final int NR = KN + PROMOTE;//「成り麒」=「獅」
public static final int SFU = SENTE + FU;    //「先手」の歩
public static final int SKY = SENTE + KY;
public static final int SGI = SENTE + GI;
public static final int SKI = SENTE + KI;
public static final int SKA = SENTE + KA;
public static final int SHI = SENTE + HI;
public static final int SOU = SENTE + OU;
public static final int SCH = SENTE + CH;
public static final int SMO = SENTE + MO;
public static final int SDO = SENTE + DO;
public static final int SKO = SENTE + KO;
public static final int SSU = SENTE + SU;
public static final int SHE = SENTE + HE;
public static final int SYO = SENTE + YO;
public static final int SSH = SENTE + SH;
```

```
public static final int SRY = SENTE + RY;
public static final int SRO = SENTE + RO;
public static final int SHO = SENTE + HO;
public static final int SHN = SENTE + HN;
public static final int SKN = SENTE + KN;
public static final int SSS = SENTE + SS;
public static final int STO = SENTE + TO;
public static final int SHA = SENTE + HA;
public static final int SNG = SENTE + NG;
public static final int SNI = SENTE + NI;
public static final int SUM = SENTE + UM;
public static final int SNH = SENTE + NH;
public static final int SNC = SENTE + NC;
public static final int SNM = SENTE + NM;
public static final int SND = SENTE + ND;
public static final int SNK = SENTE + NK;
public static final int STA = SENTE + TA;
public static final int SKE = SENTE + KE;
public static final int SNY = SENTE + NY;
public static final int SNS = SENTE + NS;
public static final int SNO = SENTE + NO;
public static final int SHJ = SENTE + HJ;
public static final int SNN = SENTE + NN;
public static final int SNR = SENTE + NR;
public static final int GFU = GOTE + FU;      //「後手」の歩
public static final int GKY = GOTE + KY;
public static final int GGI = GOTE + GI;
public static final int GKI = GOTE + KI;
public static final int GKA = GOTE + KA;
public static final int GHI = GOTE + HI;
public static final int GOU = GOTE + OU;
public static final int GCH = GOTE + CH;
public static final int GMO = GOTE + MO;
public static final int GDO = GOTE + DO;
public static final int GKO = GOTE + KO;
public static final int GSU = GOTE + SU;
public static final int GHE = GOTE + HE;
public static final int GYO = GOTE + YO;
public static final int GSH = GOTE + SH;
```

```
public static final int GRY = GOTE + RY;
public static final int GRO = GOTE + RO;
public static final int GHO = GOTE + HO;
public static final int GHN = GOTE + HN;
public static final int GKN = GOTE + KN;
public static final int GSS = GOTE + SS;
public static final int GTO = GOTE + TO;
public static final int GHA = GOTE + HA;
public static final int GNG = GOTE + NG;
public static final int GNI = GOTE + NI;
public static final int GUM = GOTE + UM;
public static final int GNH = GOTE + NH;
public static final int GNC = GOTE + NC;
public static final int GNM = GOTE + NM;
public static final int GND = GOTE + ND;
public static final int GNK = GOTE + NK;
public static final int GTA = GOTE + TA;
public static final int GKE = GOTE + KE;
public static final int GNY = GOTE + NY;
public static final int GNS = GOTE + NS;
public static final int GNO = GOTE + NO;
public static final int GHJ = GOTE + HJ;
public static final int GNN = GOTE + NN;
public static final int GNR = GOTE + NR;
public static final int WALL = 256;//盤の外を表すための定数
//先手の駒かどうかの判定
static public boolean isSente(int koma) {
        return (koma & SENTE) != 0;
}
//後手の駒かどうかの判定
static public boolean isGote(int koma) {
        return (koma & GOTE) != 0;
}
//手番から見て自分の駒かどうか判定
static public boolean isSelf(int teban, int koma) {
        if (teban == SENTE) {
                return isSente(koma);
        } else {
                return isGote(koma);
```

```
		}
}
//手番から見て相手の駒かどうか判定
static public boolean isEnemy(int teban, int koma) {
		if (teban == SENTE) {
			return isGote(koma);
		} else {
			return isSente(koma);
		}
}
//駒の種類の取得
static public int getKomashu(int koma) {
		return koma & 0x1f;
}
//駒が成れるかどうかを表す
public static final boolean canPromote[] = {

false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,//先手でも後手でもない駒
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,//先手でも後手でもない駒
false,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,
true,true,false,true,false,//空、先手の歩、香、銀、金、角、飛、仲、猛、銅、虎、酔、反、横、堅、馬、
龍、鳳、奔、麒、獅
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,//先手の王、と、白、堅、飛、馬、
龍、酔、角、横、鹿、太、鯨、猪、牛、鷹、鷲、奔、　、獅、
false, true, true, true, true, true, true, true, true, true, true, true, true, true, true, true, true, true,
false, true, false,//空、後手の歩、香、銀、金、角、飛、仲、猛、銅、虎、酔、反、横、堅、馬、龍、鳳、
奔、麒、獅
false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false, false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,//後手の王、と、白、堅、飛、馬、龍、酔、角、横、鹿、太、鯨、
猪、牛、鷹、鷲、奔、　、獅、
};
static public boolean canPromote(int koma) {
		return canPromote[koma];
}
```

```
	//駒の文字列化用の文字列
	static public final String komaString[] = { " ", "歩", "香", "銀", "金","角", "飛", "仲","猛","銅","虎","酔","反
	","横","堅","馬","龍","鳳","奔","麒","獅","王",
	"と", "白", "堅", "飛", "馬", "龍", "酔","角","横","鹿","太","鯨","猪","牛","鷹","鷲","奔","","獅","" };
	//駒の文字列化…盤面の表示用
	static public String toBanString(int koma) {
		if (koma == EMPTY) {
			return "　　";
		} else if ((koma & SENTE) != 0) {
			return " " + komaString[getKomashu(koma)];//先手の駒には" "を頭に追加
		} else {
			return "v" + komaString[getKomashu(koma)];//後手の駒には" "を頭に追加
		}
	}
	//駒の文字列化…持ち駒、手などの表示用
	static public String toString(int koma) {
		return komaString[getKomashu(koma)];
	}
}
```

Kyokumen.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
import java.util.ArrayList;
class Kyokumen implements Constants, Cloneable {
	//盤面
	int ban[][];
	//持ち駒
	ArrayList<Integer> hand[];
	//手番
	int teban = SENTE;
	public Kyokumen() {
		ban = new int[14][14];
		hand = new ArrayList[2];
		hand[0] = new ArrayList<Integer>();
		hand[1] = new ArrayList<Integer>();
	}
	//局面のコピーを行なう
	public Kyokumen clone() {
		Kyokumen k = new Kyokumen();
		//盤面のコピー
```

```
		for (int suji = 0; suji < 14; suji++) {
			for (int dan = 0; dan < 14; dan++) {
				k.ban[suji][dan] = ban[suji][dan];
			}
		}
		//持ち駒のコピー
		k.hand[0] = (ArrayList<Integer>) hand[0].clone();
		k.hand[1] = (ArrayList<Integer>) hand[1].clone();
		//手番のコピー
		k.teban = teban;
		return k;
	}
	//局面が同一かどうか
	public boolean equals(Object o) {
		Kyokumen k = (Kyokumen) o;
		if (k == null)
			return false;
		return equals(k);
	}
	//盤面の比較
	//各マスについて…
	public boolean equals(Kyokumen k) {
		if (teban != k.teban) {
			return false;
		}
		for (int suji = 1; suji <= 12; suji++) {// 盤面上の筋と段にある駒が、比較対象の盤面上の
			for (int dan = 1; dan <= 12; dan++) {//同じ位置にある駒と同じかどうか比較する。
				if (!(ban[suji][dan] == k.ban[suji][dan])) {
					return false;//違う場合は、false を返す。
				}
			}
		}
		//持ち駒の比較
		//駒の種類ごとに枚数が同じかどうかを比較する。
		int handSente[] = new int[Koma.SS + 1];
		int handGote[] = new int[Koma.SS + 1];
		int compareHandSente[] = new int[Koma.SS + 1];
		int compareHandGote[] = new int[Koma.SS + 1];
		//先手の持ち駒
```

```
        for (int i = 0; i < hand[0].size(); i++) {
                int koma = hand[0].get(i);
                int komaShu = Koma.getKomashu(koma);
                handSente[komaShu]++;
        }
        //後手の持ち駒
        for (int i = 0; i < hand[1].size(); i++) {
                int koma = hand[1].get(i);
                int komaShu = Koma.getKomashu(koma);
                handGote[komaShu]++;
        }
        //比較対象の先手の持ち駒
        for (int i = 0; i < k.hand[0].size(); i++) {
                int koma = k.hand[0].get(i);
                int komaShu = Koma.getKomashu(koma);
                compareHandSente[komaShu]++;
        }
        //比較対象の後手の持ち駒
        for (int i = 0; i < k.hand[1].size(); i++) {
                int koma = k.hand[1].get(i);
                int komaShu = Koma.getKomashu(koma);
                compareHandGote[komaShu]++;
        }
        //持ち駒の枚数を比較する
        for (int i = Koma.FU; i <= Koma.SS; i++) {
                if (handSente[i] != compareHandSente[i])
                        return false;
                if (handGote[i] != compareHandGote[i])
                        return false;
        }
        //完全に一致した
        return true;
}
//ある位置にあるコマを取得する
public int get(Position p) {
        if (p.suji < 1 || 12 < p.suji || p.dan < 1 || 12 < p.dan) {
                return Koma.WALL;//盤外なら「盤外＝壁」を返す
        }
        return ban[p.suji][p.dan];
```

```
}
//ある位置にあるコマを置く
public void put(Position p, int koma) {
	ban[p.suji][p.dan] = koma;
}
// 与えられた手で一手進めてみる
public void move(Te te) {
	//駒の行き先に駒があった場合
	if (get(te.to) != Koma.EMPTY) {
		//持ち駒にする
		if (Koma.isSente(get(te.to))) {
			//取った駒が先手の駒なら後手の持ち駒になる
			int koma = get(te.to);
			//成りなどのフラグ、先手・後手のフラグをクリア
			koma = koma & 0x0f;
			//後手の駒としてフラグをセット
			koma = koma | GOTE;
			hand[1].add(koma);
		} else {
			int koma = get(te.to);
			koma = koma & 0x0f;
			koma = koma | SENTE;
			hand[0].add(koma);
		}
	}
	if (te.from.suji == 0) {
		//持ち駒を打つ
		if (Koma.isSente(te.koma)) {
			//先手の駒の場合、先手の持ち駒を減らす。
			for (int i = 0; i < hand[0].size(); i++) {
				int koma = hand[0].get(i);
				if (koma == te.koma) {
					hand[0].remove(i);
					break;
				}
			}
		} else {
			for (int i = 0; i < hand[1].size(); i++) {
				int koma = hand[1].get(i);
```

```
                                        if (koma == te.koma) {
                                                hand[1].remove(i);
                                                break;
                                        }
                                }
                        }
                } else {
                        //盤上の駒を進めたので、元の位置は、EMPTY にする。
                        put(te.from, Koma.EMPTY);
                }
                //駒を移動先に進める
                int koma = te.koma;
                if (te.promote) {
                        //成りの処理
                        koma = koma | Koma.PROMOTE;
                }
                put(te.to, koma);
        }
        //玉を探して位置を返す
        public Position searchGyoku(int teban) {
                int toSearch = teban | Koma.OU;
                for (int suji = 1; suji <= 12; suji++) {
                        for (int dan = 1; dan <= 12; dan++) {
                                if (ban[suji][dan] == toSearch) {
                                        return new Position(suji, dan);
                                }
                        }
                }
                //見つからなかった場合、駒の利きが届かない盤外を返す
                return new Position(-2, -2);
        }
        //CSA 形式の将譜ファイル文字列
        static final String csaKomaTbl[] = {
        "","FU","KY","GI","KI","KA","HI","OU","CH","MO","DO","KO","SU","HE","YO","SH","RY","RO","HO",
        "HN","KN","SS","TO","HA","NG","NI","UM","NH","NC","NM","ND","NK","TA","KE","NY","NS","NO",
        "HJ","NN","NR","","+FU","+KY","+GI","+KI","+KA","+HI","+OU","+CH","+MO","+DO","+KO","+SU","
        +HE","+YO","+SH","+RY","+RO","+HO","+HN","+KN","+SS","+TO","+HA","+NG","+NI","+UM","+NH
        ","+NC","+NM","+ND","+NK","+TA","+KE","+NY","+NS","+NO","+HJ","+NN","+NR","","-FU","-KY","-
        GI","-KI","-KA","-HI","-OU","-CH","-MO","-DO","-KO","-SU","-HE","-YO","-SH","-RY","-RO","-HO","-H
```

```
N","-KN","-SS","-TO","-HA","-NG","-NI","-UM","-NH","-NC","-NM","-ND","-NK","-TA","-KE","-NY","-N
S","-NO","-HJ","-NN","-NR"
};
//CSA 形式の将譜ファイルから、局面を読み込む
public void ReadCsaKifu(String[] csaKifu) {
        //持ち駒
        int motigoma[][] = new int[2][Koma.SS + 1];
        //駒箱に入っている残りの駒
        int restKoma[] = new int[Koma.SS + 1];
        //持ち駒を空にする
        for (int i = 0; i <= Koma.SS; i++) {
                motigoma[0][i] = 0;
                motigoma[1][i] = 0;
        }
        //駒箱に入っている駒
        restKoma[Koma.FU] = 24;
        restKoma[Koma.KY] = 4;
        restKoma[Koma.GI] = 4;
        restKoma[Koma.KI] = 4;
        restKoma[Koma.KA] = 4;
        restKoma[Koma.HI] = 4;
        restKoma[Koma.CH] = 4;
        restKoma[Koma.MO] = 4;
        restKoma[Koma.DO] = 4;
        restKoma[Koma.KO] = 4;
        restKoma[Koma.SU] = 2;
        restKoma[Koma.HE] = 4;
        restKoma[Koma.YO] = 4;
        restKoma[Koma.SH] = 4;
        restKoma[Koma.RY] = 4;
        restKoma[Koma.RO] = 4;
        restKoma[Koma.HO] = 2;
        restKoma[Koma.HN] = 2;
        restKoma[Koma.KN] = 2;
        restKoma[Koma.SS] = 2;
        //盤面を空に初期化
        for (int suji = 1; suji <= 12; suji++) {
                for (int dan = 1; dan <= 12; dan++) {
                        ban[suji][dan] = Koma.EMPTY;
```

```
            }
}
//文字列から読み込み
for (int i = 0; i < csaKifu.length; i++) {
            String line = csaKifu[i];
            System.out.println("" + i + " :" + line);
            if (line.startsWith("P+")) {
                        if (line.equals("P+00AL")) {
                                    //残りの駒は全部先手の持ち駒
                                    for (int koma = Koma.FU; koma <= Koma.SS; koma++) {
                                                motigoma[0][koma] = restKoma[koma];
                                    }
                        } else {
                                    //先手の持ち駒
                                    for (int j = 0; j <= line.length() - 6; j += 4) {
                                                int koma = 0;
                                                String komaStr = line.substring(j + 2 + 2, j + 2 + 4);
                                                for (int k = Koma.FU; k <= Koma.SS; k++) {
                                                            if (komaStr.equals(csaKomaTbl[k])) {
                                                                        koma = k;
                                                                        break;
                                                            }
                                                }
                                                motigoma[0][koma]++;
                                    }
                        }
            } else if (line.startsWith("P-")) {
                        if (line.equals("P-00AL")) {
                                    for (int koma = Koma.FU; koma <= Koma.SS; koma++) {
                                                motigoma[1][koma] = restKoma[koma];
                                    }
                        } else {
                                    for (int j = 0; j < line.length(); j += 4) {
                                                int koma = 0;
                                                for (int k = Koma.FU; k <= Koma.SS; k++) {
                                                            if (line.substring(j + 2, j + 4).equals(
                                                                                    csaKomaTbl[k])) {
                                                                        koma = k;
                                                                        break;
```

```
                                        }
                                }
                                motigoma[1][koma]++;
                        }
                }
        } else if (line.startsWith("P")) {
                //盤面の表現
                String danStr = line.substring(1, 2);
                int dan = 0;
                try {
                        dan = Integer.parseInt(danStr);
                } catch (Exception e) {
                }
                String komaStr;
                for (int suji = 1; suji <= 12; suji++) {
                        komaStr = line.substring(2 + (12 - suji) * 3,
                                        2 + (12 - suji) * 3 + 3);
                        int koma = Koma.EMPTY;
                        for (int k = Koma.EMPTY; k <= Koma.GNR; k++) {
                                if (komaStr.equals(csaKomaTbl[k])) {
                                        koma = k;
                                        //成りのフラグを取って、残りの駒からその種類
                                        の駒を 1 枚引いておく
                                        restKoma[(Koma.getKomashu(koma)&~Koma
                                        .PROMOTE)]--;
                                        break;
                                }
                        }
                        ban[suji][dan] = koma;
                }
        } else if (line.equals("-")) {
                teban = GOTE;
        } else if (line.equals("+")) {
                teban = SENTE;
        }
}
//持ち駒を hand にしまう
for (int i = Koma.FU; i < Koma.SS; i++) {
        for (int j = 0; j < motigoma[0][i]; j++) {
```

```
                        hand[0].add(new Integer(i | SENTE));
                }
                for (int j = 0; j < motigoma[1][i]; j++) {
                        hand[1].add(new Integer(i | GOTE));
                }
        }
}
//局面を表示用に文字列化
public String toString() {
        String s = "";
        s += "後手持ち駒:";
        for (int i = 0; i < hand[1].size(); i++) {
                s += Koma.toString(hand[1].get(i));
        }
        s += "¥n";
        s += "  c   b   a   9   8   7   6   5   4   3   2   1 ¥n";
        s += "+---+---+---+---+---+---+---+---+---+---+---+---+¥n";
        for (int dan = 1; dan <= 12; dan++) {
                for (int suji = 12; suji >= 1; suji--) {
                        s += "|";
                        s += Koma.toBanString(ban[suji][dan]);
                }
                s += "|";
                s += danStr[dan];
                s += "¥n";
                s += "+---+---+---+---+---+---+---+---+---+---+---+---+¥n";
        }
        s += "先手持ち駒:";
        for (int i = 0; i < hand[0].size(); i++) {
                s += Koma.toString(hand[0].get(i));
        }
        s += "¥n";
        return s;
}
//局面を評価するための、駒の価値
static final int komaValue[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,100,600,1000,1200,1800,2000,30
        0,700,800,1400,1400,1500,1700,1700,2000,2200,2500,2800,2600,3000,10000,1200,2100,1700
        ,2000,2000,2200,1400,1800,1700,1900,1500,2100,2300,2300,3100,3200,2800,0,3000,0,0,0,0,0
```

```
            ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-100,-600,-1000,-1200,-1800,-2000,-300,-700,-800,-1400,
            -1400,-1500,-1700,-1700,-2000,-2200,-2500,-2800,-2600,-3000,-10000,-1200,-2100,-1700,-200
            0,-2000,-2200,-1400,-1800,-1700,-1900,-1500,-2100,-2300,-2300,-3100,-3200,-2800,0,-3000,0,
            0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
        };
        //局面を評価する関数
        public int evaluate() {
            int eval = 0;
            //盤面上の駒の価値を全部計算
            for (int suji = 1; suji <= 12; suji++) {
                for (int dan = 1; dan <= 12; dan++) {
                    int koma = ban[suji][dan];
                    eval = eval + komaValue[koma];
                }
            }
            //先手・後手の持ち駒の価値を全部計算
            for (int i = 0; i < 2; i++) {
                for (int j = 0; j < hand[i].size(); j++) {
                    int koma = hand[i].get(j);
                    eval = eval + komaValue[koma];
                }
            }
            return eval;
        }
}
```

KomaMoves.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
public interface KomaMoves {
        //方向の定義に沿った、「段」の移動の定義
        public static final int diffDan[]={
            1,1,1,0,0,-1,-1,-1,-2,0,0,2,-2,-2,2,2
        };
        //方向の定義に沿った、「筋」の移動の定義
        public static final int diffSuji[]={
            -1,0,1,1,-1,1,0,-1,0,2,-2,0,2,-2,2,-2
        };
```

通常の 8 方向の定義　　麒麟飛びの方向の定義　　鳳凰飛びの方向の定義

5　6　7　8　12　13

3　駒　4　9　麒　10　鳳

2　1　0　11　14　15

//ある方向に動けるかどうかを表す

//「香車」などの走り駒は後述の canJump で表すので、ここでは「false」にしておく

public static final boolean canMove[][]={

//方向０斜め右下への動き

{

false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,true,false,false,fal
se,false,false,true,false,true,true,false,false,false,false,true,false,false,true,false,true,false,fa
lse,false,false,false,true,true,false,false,true,true,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,true,true,false,false,false,true,true,true,true,fa
lse,false,false,false,true,false,false,true,false,true,true,false,false,false,false,true,true,false,f
alse,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,

},

//方向１真下への動き

{

false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,true,false,false,fal
se,false,true,true,true,true,false,false,true,false,true,false,true,false,false,false,true,true,fals
e,false,false,true,false,false,true,false,true,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,true,false,true,true,false,false,true,true,true,false,true,fals
e,true,false,true,false,true,false,false,false,true,true,false,false,false,true,false,true,false,tru
e,false,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,

},

//方向２斜め左下への動き

{

false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,true,false,fal
se,false,false,true,false,true,true,false,false,false,false,true,false,false,true,false,true,false,fa
lse,false,false,false,true,true,false,false,true,true,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,true,true,false,false,false,true,true,true,true,fa
lse,false,false,false,true,false,false,true,false,true,true,false,false,false,false,true,true,false,f
alse,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
},
//方向 3 左への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,true,fal
se,false,false,false,false,true,true,false,false,true,true,false,true,false,false,false,true,true,fal
se,true,false,true,false,true,false,false,true,true,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,true,false,false,false,false,false,true,true,fa
lse,false,true,true,false,true,false,false,false,true,true,false,true,false,true,false,true,false,fal
se,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
},
//方向 4 右への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,true,fal
se,false,false,false,false,true,true,false,false,true,true,false,true,false,false,false,true,true,fal
se,true,false,true,false,true,false,false,true,true,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,true,false,false,false,false,false,true,true,fa
lse,false,true,true,false,true,false,false,false,true,true,false,true,false,true,false,true,false,fal
se,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,

},
//方向５斜め左上への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,true,true,fal
se,false,false,true,true,true,true,false,false,false,false,true,false,false,true,false,true,true,fal
se,false,false,false,true,true,false,false,true,true,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,true,false,false,false,false,true,false,true,true,f
alse,false,false,false,true,false,false,true,false,true,false,false,false,false,false,true,true,false,
false,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
},
//方向６真上への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,true,false,true,true,fals
e,false,true,true,true,false,true,false,true,false,true,false,true,false,false,false,true,true,false
,false,false,true,false,true,false,true,false,true,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,true,false,false,true,true,true,true,false,false
,true,false,true,false,true,false,false,false,true,true,false,false,false,true,false,false,true,false
,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
},
//方向７斜め右上への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,true,true,fal
se,false,false,true,true,true,true,false,false,false,false,true,false,false,true,false,true,true,fal
se,false,false,false,true,true,false,false,true,true,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,true,false,false,false,false,true,false,true,true,f

alse,false,false,false,true,false,false,true,false,true,false,false,false,false,false,true,true,false,
false,true,true,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
},
//方向８麒麟飛び
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false
,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false
,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,
},
//方向９麒麟飛び
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,true,false,false
,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false
,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,
},
//方向１０麒麟飛び
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,

```
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false
,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false
,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,
},
//方向１１麒麟飛び
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false
,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false
,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,
},
//方向１２鳳凰飛び
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false
,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false
,false,false,false,false,false,false,true,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,
```

},
//方向１３鳳凰飛び
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false
,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false
,false,false,false,false,false,false,true,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,
},
//方向１４鳳凰飛び
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false
,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false
,false,false,false,false,false,false,true,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,
},
//方向１５鳳凰飛び
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false
,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,f

```
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false
,false,false,false,false,false,false,true,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,
}
};
static final public boolean canJump[][]={
//方向０斜め右下への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,tr
ue,false,false,false,false,false,false,false,false,false,true,false,false,true,false,true,false,false,f
alse,false,false,true,false,false,true,false,false,false,true,true,true,true,true,true,false,true,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,f
alse,false,false,true,false,false,true,false,true,false,false,true,false,false,true,false,false,true,
false,false,false,false,true,true,true,true,true,false,true,false,false,false,false,false,false,false
,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
},
//方向１真下への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,true,false,false,false,false,false,true,false,true,false,true,false,true,false,true,false,false,tr
ue,true,true,false,true,false,false,false,true,false,true,false,true,true,true,true,false,true,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,true,false,false,false,true,false,false,false,false,false,tru
e,false,true,false,true,false,true,false,true,false,false,true,true,true,false,true,false,false,fals
e,true,false,true,false,true,true,true,true,false,true,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
},
//方向２斜め左下への動き
{
```

false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,tr
ue,false,false,false,false,false,false,false,false,false,true,false,false,true,false,true,false,false,f
alse,false,false,true,false,false,true,false,false,false,true,true,true,true,true,true,false,true,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,f
alse,false,false,true,false,false,true,false,true,false,false,true,false,false,true,false,false,true,
false,false,false,false,true,true,true,true,true,false,true,false,false,false,false,false,false,false
,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
},
//方向３左への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,true,false,false,false,false,false,false,true,false,false,true,false,true,false,true,false,false,f
alse,false,true,false,true,false,false,true,false,false,false,true,false,true,true,true,false,true,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,
false,true,false,false,true,false,true,false,true,false,false,false,false,true,false,true,false,false
,true,false,false,false,true,false,true,true,true,false,true,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
},
//方向４右への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,true,false,false,false,false,false,false,true,false,false,true,false,true,false,true,false,false,f
alse,false,true,false,true,false,false,true,false,false,false,true,false,true,true,true,false,true,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,
false,true,false,false,true,false,true,false,true,false,false,false,false,true,false,true,false,false
,true,false,false,false,true,false,true,true,true,false,true,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,

```
},
//方向５斜め左上への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,tr
ue,false,false,false,false,false,false,false,false,false,true,false,false,true,false,true,false,false,
true,false,false,true,false,false,true,false,false,false,false,true,true,true,true,true,false,true,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,
false,false,false,true,false,false,true,false,true,false,false,false,false,false,true,false,false,true
,false,false,false,true,true,true,true,true,true,false,true,false,false,false,false,false,false,false
,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
},
//方向６真上への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,true,false,false,fal
se,true,false,false,false,false,false,true,false,true,false,true,false,true,false,true,false,false,tr
ue,true,true,false,true,false,false,false,true,false,true,false,true,true,true,true,false,true,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,tr
ue,false,true,false,true,false,true,false,true,false,false,true,true,true,false,true,false,false,fal
se,true,false,true,false,true,true,true,true,false,true,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
},
//方向７斜め右上への動き
{
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fa
lse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fals
e,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,tr
ue,false,false,false,false,false,false,false,false,false,true,false,false,true,false,true,false,false,
true,false,false,true,false,false,true,false,false,false,false,true,true,true,true,true,false,true,f
alse,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
se,false,false,false,false,false,false,false,false,false,false,true,false,false,false,false,false,false,
```

```
                false,false,false,true,false,false,true,false,true,false,false,false,false,false,true,false,false,true
                ,false,false,false,true,true,true,true,true,true,false,true,false,false,false,false,false,false,false
                ,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,
                }
        };
}
```

GenerateMoves.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
import java.util.ArrayList;
public class GenerateMoves implements Constants, KomaMoves {
        //各手について自分の玉に王手がかかっていないかどうかチェックし、王手のかかっている手を取り除く
        public static ArrayList<Te> removeSelfMate(Kyokumen k, ArrayList<Te> v) {
                ArrayList<Te> removed = new ArrayList<Te>();
                for (int i = 0; i < v.size(); i++) {
                        //手を取り出す
                        Te te = v.get(i);
                        //その手で 1 手進める
                        Kyokumen test = (Kyokumen) k.clone();
                        test.move(te);
                        //自玉を探す
                        Position gyokuPosition = test.searchGyoku(k.teban);
                        //王手を放置しているかどうかのフラグ
                        boolean isOuteHouchi = false;
                        //玉の周辺から相手の駒が利いている場合、その手を取り除く
                        for (int direct = 0; direct < 16 && !isOuteHouchi; direct++) {
                                Position pos = (Position) gyokuPosition.clone();
                                pos.sub(direct);
                                int koma = test.get(pos);
                                if (Koma.isEnemy(test.teban, koma) && canMove[direct][koma]) {
                                        isOuteHouchi = true;
                                        break;
                                }
                        }
                        for (int direct = 0; direct < 8 && !isOuteHouchi; direct++) {
                                Position pos = (Position) gyokuPosition.clone();
                                int koma;
                                for (pos.sub(direct), koma = test.get(pos); koma != Koma.WALL; pos
                                                .sub(direct), koma = test.get(pos)) {
                                        if (Koma.isSelf(test.teban, koma))
```

```
                                        break;
                                if (Koma.isEnemy(test.teban, koma) && canJump[direct][koma])
                                {
                                        isOuteHouchi = true;
                                        break;
                                }
                                if (Koma.isEnemy(test.teban, koma)) {
                                        break;
                                }
                        }
                }
                if (!isOuteHouchi) {
                        removed.add(te);
                }
        }
        return removed;
}
//手番、駒の種類、移動元、移動先を考慮して成る・不成りを判断しながら手を追加する
public static void addTe(ArrayList<Te> v, int teban, int koma, Position from, Position to) {
        if (teban == SENTE) {
                if ((Koma.getKomashu(koma) == Koma.KY || Koma.getKomashu(koma) ==
                Koma.FU)&& to.dan == 1) {
                        Te te = new Te(koma, from, to, true);
                        v.add(te);
                } else if ((to.dan <= 4 || from.dan <= 4) && Koma.canPromote(koma)) {
                        Te te = new Te(koma, from, to, true);
                        v.add(te);
                        te = new Te(koma, from, to, false);
                        v.add(te);
                } else {
                        Te te = new Te(koma, from, to, false);
                        v.add(te);
                }
        } else {
                if ((Koma.getKomashu(koma) == Koma.KY || Koma.getKomashu(koma) ==
                Koma.FU)&& to.dan == 12) {
                        Te te = new Te(koma, from, to, true);
                        v.add(te);
                } else if ((to.dan >= 9 || from.dan >= 9) && Koma.canPromote(koma)) {
```

```
                    Te te = new Te(koma, from, to, true);
                    v.add(te);
                    te = new Te(koma, from, to, false);
                    v.add(te);
                } else {
                    Te te = new Te(koma, from, to, false);
                    v.add(te);
                }
            }
        }
    //打ち歩詰めになっていないかどうかチェックする
    public static boolean isUtiFuDume(Kyokumen k, Te te) {
            if (te.from.suji != 0 && te.from.dan != 0) {
                return false;
            }
            if (Koma.getKomashu(te.koma) != Koma.FU) {
                return false;
            }
            int teban;
            int tebanAite;
            if ((te.koma & SENTE) != 0) {
                teban = SENTE;
                tebanAite = GOTE;
            } else {
                teban = GOTE;
                tebanAite = SENTE;
            }
            Position gyokuPositionAite = k.searchGyoku(tebanAite);
            if (teban == SENTE) {
                if (gyokuPositionAite.suji != te.to.suji || gyokuPositionAite.dan != te.to.dan - 1) {
                    return false;
                }
            } else {
                if (gyokuPositionAite.suji != te.to.suji || gyokuPositionAite.dan != te.to.dan + 1) {
                    return false;
                }
            }
            Kyokumen test = (Kyokumen) k.clone();
            test.move(te);
```

```
		test.teban = tebanAite;
		ArrayList<Te> v = generateLegalMoves(test);
		if (v.size() == 0) {
			return true;
		}
		return false;
}
//与えられた局面における合法手を生成する
public static ArrayList<Te> generateLegalMoves(Kyokumen k) {
		ArrayList<Te> v = new ArrayList<Te>();
		for (int suji = 1; suji <= 12; suji++) {
			for (int dan = 1; dan <= 12; dan++) {
				Position from = new Position(suji, dan);
				int koma = k.get(from);
				if (Koma.isSelf(k.teban, koma)) {
					for (int direct = 0; direct < 16; direct++) {
						if (canMove[direct][koma]) {
							Position to = new Position(suji +
							diffSuji[direct],dan + diffDan[direct]);
							if (1 <= to.suji && to.suji <= 12 && 1 <= to.dan
							&& to.dan <= 12) {
								if (Koma.isSelf(k.teban, k.get(to))) {
									continue;
								}
								addTe(v, k.teban, koma, from, to);
							}
						}
					}
					//各方向に「走る」手を生成
					for (int direct = 0; direct < 8; direct++) {
						if (canJump[direct][koma]) {
							for (int i = 1; i < 12; i++) {
								Position to = new Position(suji
								+ diffSuji[direct] * i, dan
								+ diffDan[direct] * i);
								if (k.get(to) == Koma.WALL)
									break;
								if (Koma.isSelf(k.teban, k.get(to)))
									break;
```

```
                                                addTe(v, k.teban, koma, from, to);
                                                if (k.get(to) != Koma.EMPTY)
                                                        break;
                                        }
                                }
                        }
                }
        }
}
booleanisPutted[]=
        {false,false,false,false,false,false,false,false,false,false,false,false,false,false,false,fal
        se,false,false,false,false,false};
ArrayList<Integer> motigoma;
if (k.teban == SENTE) {
        motigoma = k.hand[0];
} else {
        motigoma = k.hand[1];
}
for (int i = 0; i < motigoma.size(); i++) {
        int koma = motigoma.get(i);
        int komashu = Koma.getKomashu(koma);
        if (isPutted[komashu]) {
                continue;
        }
        isPutted[komashu] = true;
        for (int suji = 1; suji <= 12; suji++) {
                if (komashu == Koma.FU) {
                        boolean isNifu = false;
                        for (int dan = 1; dan <= 12; dan++) {
                                Position p = new Position(suji, dan);
                                if (k.get(p) == (k.teban | Koma.FU)) {
                                        isNifu = true;
                                        break;
                                }
                        }
                        if (isNifu) {
                                continue;
                        }
                }
```

```
                                for (int dan = 1; dan <= 12; dan++) {
                                        if (komashu == Koma.FU || komashu == Koma.KY) {
                                                if (k.teban == SENTE && dan == 1) {
                                                        continue;
                                                } else if (k.teban == GOTE && dan == 12) {
                                                        continue;
                                                }
                                        }
                                        Position from = new Position(0, 0);
                                        Position to = new Position(suji, dan);
                                        if (k.get(to) != Koma.EMPTY) {
                                                continue;
                                        }
                                        Te te = new Te(koma, from, to, false);
                                        if (isUtiFuDume(k, te)) {
                                                continue;
                                        }
                                        v.add(te);
                                }
                        }
                }
                return v;
        }
}
```

Human.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
import java.util.ArrayList;
import java.io.*;
public class Human implements Player, Constants {
        static BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        public Te getNextTe(Kyokumen k) {
                //現在の局面での合法手を生成
                ArrayList<Te> v = GenerateMoves.generateLegalMoves(k);
                Position toryoPos = new Position(0, 0);
                Te te = new Te(0, toryoPos, toryoPos, false);

                do {
                        if (k.teban == SENTE) {
                                System.out.println("先手番です。");
```

```
} else {
		System.out.println("後手番です。");
}
System.out.println("指し手を入力してください。");
String s = "";
try {
		s = reader.readLine();
} catch (Exception e) {
		e.printStackTrace();
		break;
}
if (s.equals("%TORYO")) {//投了
		break;
}
if (s.equals("p")) {//合法手の一覧を表示
		for (int i = 0; i < v.size(); i++) {
			Te t = v.get(i);
			System.out.println(t);
		}
		System.out.println(k);
		continue;
}
if(s.length() > 5){
		System.out.println("入力が異常です。");
		System.out.println(k);
		continue;
}
boolean promote = false;
if (s.length() == 5) {
		if (s.substring(4, 5).equals("*")) {//5 文字目に*を入力すると成る
			promote = true;
		} else {
			System.out.println("入力が異常です。");
			System.out.println(k);
			continue;
		}
}
int fromSuji = 0, fromDan = 0, toSuji = 0, toDan = 0;
try {
```

```
                    toDan = Integer.parseInt(s.substring(3, 4));
                    fromSuji = Integer.parseInt(s.substring(0, 1), 16);
                    fromDan = Integer.parseInt(s.substring(1, 2), 16);
                    toSuji = Integer.parseInt(s.substring(2, 3), 16);
                    toDan = Integer.parseInt(s.substring(3, 4), 16);
                } catch (Exception e) {
                    System.out.println("手を読み込めませんでした。");
                    System.out.println("" + fromSuji + "" + fromDan + "" + toSuji
                    + "" + toDan);
                    System.out.println(k);
                    continue;
                }
                int koma = 0;
                if (fromSuji == 0) {
                    koma = fromDan | k.teban;
                    fromDan = 0;
                }
                Position from = new Position(fromSuji, fromDan);
                Position to = new Position(toSuji, toDan);
                if (fromSuji != 0) {
                    koma = k.get(from);
                }
                te = new Te(koma, from, to, promote);
                if (!v.contains(te)) {
                    System.out.println(te);
                    System.out.println("合法手ではありません。");
                    System.out.println(k);
                }
            } while (!v.contains(te));
            return te;
        }
}
```

Main.java のソースコード

```
package jp.usapyonsoft.lesserpyon;

import java.util.ArrayList;
import java.io.*;
public class Main implements Constants {
        //初期盤面を与える
```

```
static final int ShokiBanmen[][] = {
        { Koma.GKY, Koma.GMO, Koma.GDO, Koma.GGI, Koma.GKI, Koma.GSU, Koma.GOU,
        Koma.GKI, Koma.GGI, Koma.GDO, Koma.GMO, Koma.GKY},{ Koma.GHE, Koma.EMP,
        Koma.GKA, Koma.EMP, Koma.GKO, Koma.GHO, Koma.GKN, Koma.GKO, Koma.EMP,
        Koma.GKA, Koma.EMP, Koma.GHE},{ Koma.GYO, Koma.GSH, Koma.GHI, Koma.GRY,
        Koma.GRO, Koma.GHN, Koma.GSS, Koma.GRO, Koma.GRY, Koma.GHI, Koma.GSH,
        Koma.GYO},{ Koma.GFU, Koma.GFU, Koma.GFU, Koma.GFU, Koma.GFU, Koma.GFU,
        Koma.GFU, Koma.GFU, Koma.GFU, Koma.GFU, Koma.GFU, Koma.GFU},{ Koma.EMP,
        Koma.EMP, Koma.EMP, Koma.GCH, Koma.EMP, Koma.EMP, Koma.EMP, Koma.EMP,
        Koma.GCH, Koma.EMP, Koma.EMP, Koma.EMP},{ Koma.EMP, Koma.EMP, Koma.EMP,
        Koma.EMP, Koma.EMP, Koma.EMP, Koma.EMP, Koma.EMP, Koma.EMP, Koma.EMP,
        Koma.EMP, Koma.EMP}, { Koma.EMP, Koma.EMP, Koma.EMP, Koma.EMP, Koma.EMP,
        Koma.EMP,Koma.EMP,Koma.EMP,Koma.EMP,Koma.EMP,Koma.EMP,Koma.EMP},{ Koma.
        EMP, Koma.EMP, Koma.EMP, Koma.SCH, Koma.EMP, Koma.EMP, Koma.EMP, Koma.EMP,
        Koma.SCH, Koma.EMP, Koma.EMP, Koma.EMP},{ Koma.SFU, Koma.SFU, Koma.SFU,
        Koma.SFU, Koma.SFU, Koma.SFU, Koma.SFU, Koma.SFU, Koma.SFU, Koma.SFU,
        Koma.SFU, Koma.SFU},{ Koma.SYO, Koma.SSH, Koma.SHI, Koma.SRY, Koma.SRO,
        Koma.SSS, Koma.SHN, Koma.SRO, Koma.SRY, Koma.SHI, Koma.SSH,
        Koma.SYO},{ Koma.SHE, Koma.EMP, Koma.SKA, Koma.EMP, Koma.SKO, Koma.SKN,
        Koma.SHO, Koma.SKO, Koma.EMP, Koma.SKA, Koma.EMP, Koma.SHE},{ Koma.SKY,
        Koma.SMO, Koma.SDO, Koma.SGI, Koma.SKI, Koma.SOU, Koma.SSU, Koma.SKI,
        Koma.SGI, Koma.SDO, Koma.SMO, Koma.SKY},
        };
static Player player[]=new Player[2];
static ArrayList<Kyokumen> kyokumenRireki = new ArrayList<Kyokumen>();
//使い方表示
static void usage(){
        System.out.println("使い方:");
        System.out.println("例:先手　人間　後手　コンピュータの場合");
        System.out.println("java jp.usapyonsoft.lesserpyon.Main HUMAN CPU");
        System.out.println("");
        System.out.println("初期局面を与えて対局開始することも可能です。");
        System.out.println("例:kyokumen.csa に初期局面が入っているとした場合");
        System.out.println("java jp.usapyonsoft.lesserpyon.Main HUMAN CPU kyokumen.csa");
}
//メイン関数　引数は先手が誰か、後手は誰か
public static void main(String argv[]) {
        if(argv.length<2){
                usage();
```

```
            return;
}
if(argv[0].equals("HUMAN")){
            player[0]=new Human();
}else if(argv[0].equals("CPU")){
            player[0]=new Sikou();
}else{
            usage();
            return;
}
if(argv[1].equals("HUMAN")){
            player[1]=new Human();
}else if(argv[1].equals("CPU")){
            player[1]=new Sikou();
}else{
            usage();
            return;
}
try {
            Kyokumen k = new Kyokumen();
            if (argv.length == 2) {
                        k.teban = SENTE;
                        for (int dan = 1; dan <= 12; dan++) {
                                    for (int suji = 12; suji >= 1; suji--) {
                                                k.ban[suji][dan] = ShokiBanmen[dan - 1][12 - suji];
                                    }
                        }
            } else {
                        String csaFileName = argv[2];
                        File f = new File(csaFileName);
                        BufferedReader in = new BufferedReader(new FileReader(f));
                        ArrayList<String> v = new ArrayList<String>();
                        String s;
                        while ((s = in.readLine()) != null) {
                                    System.out.println("Read:" + s);
                                    v.add(s);
                        }
                        String csaKifu[] = new String[v.size()];
                        v.toArray(csaKifu);
```

```
            k.ReadCsaKifu(csaKifu);
        }
        while(true){
            kyokumenRireki.add(k.clone());
            ArrayList<Te> v= GenerateMoves.generateLegalMoves(k);
            if(v.size()==0){
                if(k.teban==SENTE){
                    System.out.println("後手の勝ち！");
                }else{
                    System.out.println("先手の勝ち！");
                }
                break;
            }
            int sameKyokumen=0;
            for(int i=0;i<kyokumenRireki.size();i++){
                if(kyokumenRireki.get(i).equals(k)){
                    sameKyokumen++;
                }
            }
            if(sameKyokumen>=4){
                System.out.println("千日手です。");
                break;
            }
            System.out.println(k.toString());
            Te te;
            if(k.teban==SENTE){
                te=player[0].getNextTe(k);
            }else{
                te=player[1].getNextTe(k);
            }
            System.out.println(te.toString());
            if(!v.contains(te)){
                System.out.println("合法手でない手が指されました。");
                if(k.teban==SENTE){
                    System.out.println("後手の勝ち！");
                }else{
                    System.out.println("先手の勝ち！");
                }
                break;
```

```
                    }
                    k.move(te);
                    if(k.teban==SENTE){
                        k.teban=GOTE;
                    }else{
                        k.teban=SENTE;
                    }
                }
                System.out.println("対局終了です。");
                System.out.println("最後の局面は…");
                System.out.println(k.toString());
                ArrayList<Te> v = GenerateMoves.generateLegalMoves(k);
                System.out.println("可能手:" + v.size() + "手");
                for (int i = 0; i < v.size(); i++) {
                    Te te = v.get(i);
                    System.out.println(te.toString());
                }
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
}
```

Player.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
public interface Player {
    Te getNextTe(Kyokumen k);
}
```

Position.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
//駒の位置を表すクラス
class Position implements Cloneable, KomaMoves {
    public int suji;
    public int dan;
    public Position() {
        suji = 0;
        dan = 0;
    }
    public Position(int _suji, int _dan) {
        suji = _suji;
```

```
                dan = _dan;
        }
        //同一性比較用メソッド
        public boolean equals(Position p) {
                return (p.suji == suji && p.dan == dan);
        }
        public boolean equals(Object o) {
                Position p = (Position) o;
                if (p == null)
                        return false;
                return equals(p);
        }
        //コピーを返す
        public Object clone() {
                return new Position(suji, dan);
        }
        //ある方向への動きを行なう
        public void add(int diffSuji, int diffDan) {
                suji += diffSuji;
                dan += diffDan;
        }
        //ある方向への逆の動きを行なう
        public void sub(int diffSuji, int diffDan) {
                suji -= diffSuji;
                dan -= diffDan;
        }
        public void add(int direct) {
                add(diffSuji[direct], diffDan[direct]);
        }
        public void sub(int direct) {
                sub(diffSuji[direct], diffDan[direct]);
        }
}
```

Te.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
public class Te implements Cloneable, Constants {
        int koma;
        Position from;//動く前の位置
        Position to;//動いた先の位置
```

```
	boolean promote;//成る場合、true　成らない場合、false
	public Te(int _koma, Position _from, Position _to, boolean _promote) {
		koma = _koma;
		from = (Position) _from.clone();
		to = (Position) _to.clone();
		promote = _promote;
	}
	public boolean equals(Te te) {
		return (te.koma == koma && te.from.equals(from) && te.to.equals(to) && te.promote ==
		promote);
	}
	public boolean equals(Object _te) {
		Te te = (Te) _te;
		if (te == null)
			return false;
		return equals(te);
	}
	public Object clone() {
		return new Te(koma, from, to, promote);
	}
	//手を文字列で表示
	public String toString() {
		return sujiStr[to.suji]+ danStr[to.dan]+ Koma.toString(koma)+ (promote ? "成" : "")
			+ (from.suji == 0 ? "打  " : "(" + sujiStr[from.suji]+ danStr[from.dan] + ")")
			+ (promote ? "" : "    ");
	}
}
```

Sikou.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
import java.util.ArrayList;
import java.util.Random;
//コンピュータの思考ルーチン
public class Sikou implements Player, Constants {
	static final int INFINITE = 99999999;
	static final int DEPTH_MAX = 4;//読みの深さ
	static final int LIMIT_DEPTH = 16;
	//最善手順を格納する
	Te best[][] = new Te[LIMIT_DEPTH][LIMIT_DEPTH];
	int leaf = 0;
```

```
int node = 0;
public Sikou() {
}
int negaMax(Te t, Kyokumen k, int alpha, int beta, int depth, int depthMax) {
        if (depth >= depthMax) {
                leaf++;
                if (k.teban == SENTE) {
                        return k.evaluate();
                } else {
                        return -k.evaluate();
                }
        }
        node++;
        ArrayList<Te> v = GenerateMoves.generateLegalMoves(k);
        int value = -INFINITE;
        //合法手の中から最善手を選択
        for (int i = 0; i < v.size(); i++) {
                Te te = v.get(i);
                Kyokumen nextKyokumen = (Kyokumen) k.clone();
                nextKyokumen.move(te);
                if (nextKyokumen.teban == SENTE) {
                        nextKyokumen.teban = GOTE;
                } else {
                        nextKyokumen.teban = SENTE;
                }
                Te tmpTe = new Te(0, new Position(0, 0), new Position(0, 0), false);
                int eval = -negaMax(tmpTe, nextKyokumen, -beta, -alpha, depth + 1,
                                depthMax);
                if (eval > value) {
                        value = eval;
                        if (eval > alpha) {
                                alpha = eval;
                        }
                        best[depth][depth] = te;
                        t.koma = te.koma;
                        t.from = te.from;
                        t.to = te.to;
                        t.promote = te.promote;
                        for (int j = depth + 1; j < depthMax; j++) {
```

```
                                        best[depth][j] = best[depth + 1][j];
                                }
                                if (eval >= beta) {
                                        break;
                                }
                        }
                }
                return value;
        }
        public Te getNextTe(Kyokumen k) {
                leaf = node = 0;
                Te te = new Te(0, new Position(0, 0), new Position(0, 0), false);
                long time = System.currentTimeMillis();
                negaMax(te, k, -INFINITE, INFINITE, 0, DEPTH_MAX);
                time = System.currentTimeMillis() - time;
                System.out.println("leaf=" + leaf + " node=" + node + " time=" + time+ "ms");
                return te;
        }
}
```

SikouMinMax.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
import java.util.ArrayList;
import java.util.Random;
public class SikouMinMax implements Player, Constants {
        static final int INFINITE = 9999999;
        static final int DEPTH_MAX = 4;
        int leaf = 0;
        int node = 0;
        Random random;
        public SikouMinMax() {
                random = new Random();
        }
        int getMax(Te t, Kyokumen k, int depth, int depthMax) {
                if (depth >= depthMax) {
                        leaf++;
                        return k.evaluate();
                }
                node++;
                ArrayList<Te> v = GenerateMoves.generateLegalMoves(k);
```

```
        ArrayList<Te> v2 = new ArrayList<Te>();
        int value = -INFINITE;
        for (int i = 0; i < v.size(); i++) {
                Te te = v.get(i);
                Kyokumen nextKyokumen = (Kyokumen) k.clone();
                nextKyokumen.move(te);
                nextKyokumen.teban = GOTE;
                Te tmpTe = new Te(0, new Position(0, 0), new Position(0, 0), false);
                int eval = getMin(tmpTe, nextKyokumen, depth + 1, depthMax);
                if (eval > value) {
                        value = eval;
                        v2.clear();
                        v2.add(te);
                }
                if (eval == value) {
                        v2.add(te);
                }
        }
        if (v2.size() == 0) {
                return value;
        }
        Te te = v2.get(random.nextInt(v2.size()));
        t.koma = te.koma;
        t.from = te.from;
        t.to = te.to;
        t.promote = te.promote;
        return value;
}
int getMin(Te t, Kyokumen k, int depth, int depthMax) {
        if (depth >= depthMax) {
                leaf++;
                return k.evaluate();
        }
        node++;
        ArrayList<Te> v = GenerateMoves.generateLegalMoves(k);
        ArrayList<Te> v2 = new ArrayList<Te>();
        int value = INFINITE;
        for (int i = 0; i < v.size(); i++) {
                Te te = v.get(i);
```

```
                        Kyokumen nextKyokumen = (Kyokumen) k.clone();
                        nextKyokumen.move(te);
                        nextKyokumen.teban = SENTE;
                        Te tmpTe = new Te(0, new Position(0, 0), new Position(0, 0), false);
                        int eval = getMax(tmpTe, nextKyokumen, depth + 1, depthMax);
                        if (eval < value) {
                                value = eval;
                                v2.clear();
                                v2.add(te);
                        }
                        if (eval == value) {
                                v2.add(te);
                        }
                }
                if (v2.size() == 0) {
                        return value;
                }
                Te te = v2.get(random.nextInt(v2.size()));
                t.koma = te.koma;
                t.from = te.from;
                t.to = te.to;
                t.promote = te.promote;
                return value;
        }
        public Te getNextTe(Kyokumen k) {
                leaf = node = 0;
                Te te = new Te(0, new Position(0, 0), new Position(0, 0), false);
                long time = System.currentTimeMillis();
                if (k.teban == SENTE) {
                        getMax(te, k, 0, DEPTH_MAX);
                } else {
                        getMin(te, k, 0, DEPTH_MAX);
                }
                time = System.currentTimeMillis() - time;
                System.out.println("leaf=" + leaf + " node=" + node + " time=" + time+ "ms");
                return te;
        }
}
```

SikouAlphaBeta.java のソースコード

```
package jp.usapyonsoft.lesserpyon;
import java.util.ArrayList;
import java.util.Random;
public class SikouAlphaBeta implements Player, Constants {
	static final int INFINITE = 99999999;
	static final int DEPTH_MAX = 4;
	static final int LIMIT_DEPTH = 16;
	Te best[][] = new Te[LIMIT_DEPTH][LIMIT_DEPTH];
	int leaf = 0;
	int node = 0;
	public SikouAlphaBeta() {
	}
	int getMax(Te t, Kyokumen k, int alpha, int beta, int depth, int depthMax) {
		if (depth >= depthMax) {
			leaf++;
			return k.evaluate();
		}
		node++;
		ArrayList<Te> v = GenerateMoves.generateLegalMoves(k);
		int value = -INFINITE;
		for (int i = 0; i < v.size(); i++) {
			Te te = v.get(i);
			Kyokumen nextKyokumen = (Kyokumen) k.clone();
			nextKyokumen.move(te);
			nextKyokumen.teban = GOTE;
			Te tmpTe = new Te(0, new Position(0, 0), new Position(0, 0), false);
			int eval = getMin(tmpTe, nextKyokumen, alpha, beta, depth + 1,
					depthMax);
			if (eval > value) {
				value = eval;
				if (eval > alpha) {
					alpha = eval;
				}
				best[depth][depth] = te;
				t.koma = te.koma;
				t.from = te.from;
				t.to = te.to;
				t.promote = te.promote;
				for (int j = depth + 1; j < depthMax; j++) {
```

```
                                        best[depth][j] = best[depth + 1][j];
                                }
                                if (eval >= beta) {
                                        break;
                                }
                        }
                }
                return value;
}
int getMin(Te t, Kyokumen k, int alpha, int beta, int depth, int depthMax) {
                if (depth >= depthMax) {
                        leaf++;
                        return k.evaluate();
                }
                node++;
                ArrayList<Te> v = GenerateMoves.generateLegalMoves(k);
                ArrayList<Te> v2 = new ArrayList<Te>();
                int value = INFINITE;
                for (int i = 0; i < v.size(); i++) {
                        Te te = v.get(i);
                        Kyokumen nextKyokumen = (Kyokumen) k.clone();
                        nextKyokumen.move(te);
                        nextKyokumen.teban = SENTE;
                        Te tmpTe = new Te(0, new Position(0, 0), new Position(0, 0), false);
                        int eval = getMax(tmpTe, nextKyokumen, alpha, beta, depth + 1,depthMax);
                        if (eval < value) {
                                value = eval;
                                if (eval < beta) {
                                        beta = eval;
                                }
                                best[depth][depth] = te;
                                t.koma = te.koma;
                                t.from = te.from;
                                t.to = te.to;
                                t.promote = te.promote;
                                for (int j = depth + 1; j < depthMax; j++) {
                                        best[depth][j] = best[depth + 1][j];
                                }
                                if (eval <= alpha) {
```

```
                                        break;
                                }
                        }
                }
                return value;
        }
        public Te getNextTe(Kyokumen k) {
                leaf = node = 0;
                Te te = new Te(0, new Position(0, 0), new Position(0, 0), false);
                long time = System.currentTimeMillis();
                if (k.teban == SENTE) {
                        System.out.println("評価:"
                        + getMax(te, k, -INFINITE, INFINITE, 0, DEPTH_MAX));
                } else {
                        System.out.println("評価:"
                        + getMin(te, k, -INFINITE, INFINITE, 0, DEPTH_MAX));
                }
                time = System.currentTimeMillis() - time;
                System.out.println("leaf=" + leaf + " node=" + node + " time=" + time+ "ms");
                return te;
        }
}
```