

連珠 (CUI - java)

S.Matoike

目次

0.1	連珠 (Connect Four)	1
0.1.1	定数定義クラスと主処理	2
0.1.2	ゲームクラス	3
0.1.3	石クラス	4
0.1.4	盤面クラス	4
0.1.5	プレイヤークラス	7
0.1.6	戦略クラス (AlphaBeta)	8
	参考文献	11

0.1 連珠 (Connect Four)

スタート! [5 目並べ]

```
      0 1 2 3 4 5 6 7 8 9
0:  . . . . . . . . .
1:  . . . . . . . . .
2:  . . . . . . . . .
3:  . . . . . . . . .
4:  . . . . . . . . .
5:  . . . . . . . . .
6:  . . . . . . . . .
7:  . . . . . . . . .
8:  . . . . . . . . .
9:  . . . . . . . . .
```

石を置く場所 xy を指定 (あなた の番です):44

```
      0 1 2 3 4 5 6 7 8 9
0:  . . . . . . . . .
1:  . . . . . . . . .
2:  . . . . . . . . .
3:  . . . . . . . . .
4:  . . . . ● . . . .
5:  . . . . . . . . .
6:  . . . . . . . . .
7:  . . . . . . . . .
8:  . . . . . . . . .
9:  . . . . . . . . .
```

石を置く場所 xy を指定 (あなた の番です):43

(*** 中略 ***)

石を置く場所 xy を指定 (あなた の番です):53

```
      0 1 2 3 4 5 6 7 8 9
0:  . . . . . . . . .
1:  . . . . . . . . .
```

```

2:  . . . . .
3:  . . ● ● ● . .
4:  . . . ○ ● . .
5:  . . . ○ ○ ○ . .
6:  . . . . . ○ . .
7:  . . . . . ○ . .
8:  . . . . . . ● .
9:  . . . . .

```

石を置く場所 xy を指定 (あなた の番です): 36

```

      0 1 2 3 4 5 6 7 8 9
0:  . . . . .
1:  . . . . .
2:  . . . . .
3:  . . ● ● ● ● . .
4:  . . . ○ ● . .
5:  . . . ○ ○ ○ . .
6:  . . . . . ○ . .
7:  . . . . . ○ . .
8:  . . . . . . ● .
9:  . . . . .

```

'X' の勝ち またね!

0.1.1 定数定義クラスと主処理

ソースコード 1 定数定義クラス: N 目並べ

```

1  package nmoku;
2
3  public final class Constants {
4      private Constants() {}
5      final static int WIDTH = 7;
6      final static int NMOKU = 5;
7      final static int NxN = NMOKU * NMOKU;
8      final static int WHITE = 1;
9      final static int BLACK = -1;
10     final static int MARU = WHITE;
11     final static int BATSU = BLACK;
12     final static int MAX = WHITE;
13     final static int MIN = BLACK;
14     final static int EMPTY = 0;
15     final static int NEXT = 200;
16     final static int DRAW = 100;
17     final static int PROMPT = 1000;
18     final static int RANDOM = 1010;
19     final static int MINIMAX = 1001;
20     final static int ALPHABETA = 1002;
21     final static int MONTECARLO = 1003;
22 }

```

ソースコード 2 主処理: N 目並べ

```

1  package nmoku;
2
3  public class NMoku {
4      public static void main(String[] args) {

```

```
5         boolean reverse = false;
6         if (args.length > 0) {
7             if (args[0].equals("-r")) {
8                 reverse = true;
9             }
10        }
11        Game g = new Game(reverse);
12        g.Start();
13    }
14 }
```

0.1.2 ゲームクラス

ソースコード 3 ゲームクラス：N 目並べ

```
1 package nmoku;
2
3 import static nmoku.Constants.*;
4
5 public class Game {
6
7     private Player[] player = null;
8     private Board board = null;
9     private int current_player = 0;
10    private Player currPlayer = null;
11
12    public Game(boolean reverse) {
13        player = new Player[2];
14        if (reverse) {
15            player[0] = new Player(MARU, "PC", ALPHABETA);
16            player[1] = new Player(BATSU, "あなた", PROMPT);
17        } else {
18            player[0] = new Player(BATSU, "あなた", PROMPT);
19            player[1] = new Player(MARU, "PC", ALPHABETA);
20        }
21        currPlayer = player[0];
22        board = new Board();
23    }
24    void Start() {
25        int winner = NEXT;
26        System.out.println("スタート! [ " + NMOKU + " 目並べ ]");
27        do {
28            board.print();
29            currPlayer = player[current_player];
30            currPlayer.putStone(board);
31            winner = board.check();
32            current_player = ++current_player % 2;
33        } while (winner == NEXT);
34        board.print();
35        result(winner);
36    }
37    private void result(int winner) {
38        // 結果の表示
39    }
40 }
```

ソースコード 4 結果の表示：ゲームクラス内：N 目並べ

```
1 private void result(int winner) {
2     // 結果の表示
3     System.out.println();
4     switch (winner) {
5         case DRAW:
6             System.out.print("引き分け\t");
7             break;
8         case MARU:
9             System.out.print(currPlayer.getName() + "'O' の勝ち\t");
10            break;
11        case BATSU:
12            System.out.print(currPlayer.getName() + "'X' の勝ち\t");
13            break;
14    }
15    System.out.println("またね!");
16 }
```

0.1.3 石クラス

ソースコード 5 石クラス：N 目並べ

```
1 package nmoku;
2
3 import static nmoku.Constants.*;
4
5 public class Stone {
6
7     private int locate = 0;
8     private int color = EMPTY;
9
10    public Stone(int n, int i) {
11        locate = n;
12        color = i;
13    }
14    int getColor() {
15        return color;
16    }
17    void setColor(int i) {
18        color = i;
19    }
20    int getLocate() {
21        return locate;
22    }
23 }
```

0.1.4 盤面クラス

ソースコード 6 盤面クラス：N 目並べ

```
1 package nmoku;
2
3 import static nmoku.Constants.*;
4
5 public class Board {
6
7     private static Stone[] board = new Stone[WIDTH * WIDTH];
8     private static Stone recentStone = null;
9
10    public Board() {
11        for (int i = 0; i < WIDTH * WIDTH; i++) {
12            board[i] = new Stone(i, EMPTY);
13        }
14    }
15    void setBoard(Stone s) {
16        recentStone = s;
17        board[s.getLocate()] = s;
18    }
19    void setEmpty(int xy){
20        board[xy]=new Stone(xy, EMPTY);
21    }
22    boolean canPut(Stone s) {
23        if (board[s.getLocate()].getColor() != EMPTY) {
24            return false;
25        }
26        return true;
27    }
28    int getz(int x, int y) {
29        return y * WIDTH + x;    // 0<= x <WIDTH, 0<= y <WIDTH
30    }
31    int check() {
32        return boardScan(recentStone.getLocate() % WIDTH,
33                           recentStone.getLocate() / WIDTH);
34    }
35    void print() {
36        // 盤面の表示印刷
37    }
38    private int boardScan(int x, int y) {
39        //盤面の調査(5個並んだかの調査)
40    }
41    private int boardScanSub(int x, int y, int move_x, int move_y) {
42        // 盤面調査の下請け
43    }
44    boolean isWin(){
45        if(check()!=NEXT)return true;
46        return false;
47    }
48    boolean isDraw(){
49        return false;
50    }
51    float evaluate(Player player, int turn){
52        if(isWin()){
53            if(turn==player.getColor())return (float)(-1.0);
54            else if(turn!=player.getColor())return (float)(1.0);
55        }
56        return (float)0.0;
57    }
58 }
```

ソースコード 7 盤面の表示印刷：盤面クラス内：N 目並べ

```

1 void print() {
2     // 盤面の表示印刷
3     final String[] str = {".", "●", "○"};
4
5     System.out.printf("\n      ");
6     for (int x = 0; x < WIDTH; x++) {
7         if (x < 10) {
8             System.out.printf("%2d", x);
9         } else {
10            System.out.printf("%2c", (char) ('a' - 10 + x));
11        }
12    }
13    System.out.printf("\n");
14    for (int y = 0; y < WIDTH; y++) {
15        if (y < 10) {
16            System.out.printf("%2d: ", y);
17        } else {
18            System.out.printf("%2c: ", (char) ('a' - 10 + y));
19        }
20        for (int x = 0; x < WIDTH; x++) {
21            int cl = board[getz(x, y)].getColor();
22            int num=0;
23            if(cl==WHITE)num=2;
24            if(cl==BLACK)num=1;
25            System.out.printf("%2s", str[num]);
26        }
27        System.out.printf("\n");
28    }
29 }

```

ソースコード 8 盤面の調査：盤面クラス内：N 目並べ

```

1 private int boardScan(int x, int y) {
2     //盤面の調査 (5個 並んだかの調査)
3     int[] n = new int[4];        //8方向 (直線4本分) に並んだ数
4     //[ \ ] 方向
5     n[0] = boardScanSub(x, y, 1, 1);
6     //[ | ] 方向
7     n[1] = boardScanSub(x, y, 0, 1);
8     //[ - ] 方向
9     n[2] = boardScanSub(x, y, 1, 0);
10    //[ / ] 方向
11    n[3] = boardScanSub(x, y, -1, 1);
12    for (int i = 0; i < 4; i++) {
13        if (n[i] == NMOKU) {
14            return recentStone.getColor();
15        }
16    }
17    return NEXT;
18 }
19 private int boardScanSub(int x, int y, int move_x, int move_y) {
20     // 盤面調査の下請け
21     int n = 1;                    //置いた場所の1個分で初期化
22     int i;
23     for (i = 1; i < NMOKU; i++) {

```

```
24     int z = getz(x + (move_x * i), y + (move_y * i));
25     if (!(0 <= z && z < WIDTH * WIDTH)) continue;
26     if (board[z].getColor() == recentStone.getColor()) {
27         n += 1;
28     } else {
29         break;
30     }
31 }
32 for (i = 1; i < NMOKU; i++) {
33     int z = getz(x + (-1 * move_x * i), y + (-1 * move_y * i));
34     if (!(0 <= z && z < WIDTH * WIDTH)) continue;
35     if (board[z].getColor() == recentStone.getColor()) {
36         n += 1;
37     } else {
38         break;
39     }
40 }
41 return n;
42 }
```

0.1.5 プレイヤークラス

ソースコード 9 プレイヤークラス：N 目並べ

```
1 package nmoku;
2
3 import static nmoku.Constants.*;
4
5 public class Player extends Strategy {
6
7     private String Name = null;
8     private int Senryaku = 0;
9     private int Color = EMPTY;
10
11     public Player(int i, String n, int s) {
12         Color = i;
13         Name = n;
14         Senryaku = s;
15     }
16     void putStone(Board board) {
17         Stone s = null;
18         do {
19             int te = Te(board);
20             s = new Stone(te, Color);
21             if (board.canPut(s)) {
22                 break;
23             }
24         } while (true);
25         board.setBoard(s);
26     }
27     private int Te(Board board) {
28         int v = 0;
29         switch (Senryaku) {
30             case PROMPT:
31                 v = prompt(Name);
32                 break;
```



```
33         case RANDOM:
34             v = random(Name);
35             break;
36         case MINIMAX:
37             v = bestMoveMM(board, this);
38             break;
39         case ALPHABETA:
40             v = bestMoveAB(board, this);
41             break;
42         case MONTECARLO:
43             break;
44     }
45     return v;
46 }
47 int getColor(){
48     return Color;
49 }
50 String getName(){
51     return Name;
52 }
53 }
```

0.1.6 戦略クラス (AlphaBeta)

盤面の大きさが大きければ大きいほど、また MINIMAX の探索深さが深くなるほど、コンピュータの手を見つけ出すのに時間がかかってしまい、人間との対戦の場面では現実的ではありません。そこで MINIMAX を改良した、AlphaBeta という戦略を使います。

コンピュータが実用的な対戦相手となるためには、盤面サイズを小さくしたり、探索の深さを妥協したりというチューニングが必要になります。

ソースコード 10 戦略クラス：N 目並べ

```
1 package nmoku;
2
3 import static nmoku.Constants.*;
4 import java.io.BufferedReader;
5 import java.io.InputStreamReader;
6
7 public class Strategy {
8
9     private boolean numP(char x) {
10         if (('0' <= x) && (x <= '9')) {
11             return true;
12         }
13         return false;
14     }
15     private int fromA(char x) {
16         return (int) (x - 'a');
17     }
18     int prompt(String name) {
19         // 標準入力から
20     }
21     int random(String name) {
22         System.out.printf("。。。 %s 思考中 。。。", name);
23         return (int) (Math.random() * (WIDTH * WIDTH));
```

```
24     }
25     private float minimax(Board board, int depth, int turn, Player player) {
26         // 三目並べ(TicTacToe)の時と同様に
27     }
28     int bestMoveMM(Board board, Player player) {
29         // 三目並べ(TicTacToe)の時と同様に
30     }
31     private float alphabeta(Board board, int depth, int turn, Player player, float
32         alpha, float beta) {
33
34         if (board.isWin() || board.isDraw() || depth == 0) {
35             float score = board.evaluate(player, turn);
36             //System.out.println("evaluated score="+score+", depth="+depth);
37             return score;
38         }
39
40         for (int i = 0; i < WIDTH * WIDTH; i++) {
41             Stone s = new Stone(i, turn);
42             if (board.canPut(s)) {
43
44                 //board.print();
45                 //if(turn==MARU)System.out.println("Turn=O");
46                 //else System.out.println("Turn=X");
47
48                 board.setBoard(s);
49                 float score = alphabeta(board, depth - 1, -turn, player, alpha,
50                     beta);
51                 board.setEmpty(i);
52                 //System.out.println("k1="+i+", score="+score);
53
54                 if (turn == MAX) {
55                     alpha = Float.max(score, alpha);
56                     if( beta <= alpha )break;
57                 } else {
58                     beta = Float.min(score, beta);
59                     if( beta <= alpha )break;
60                 }
61             }
62         }
63         if(turn==MAX)return alpha;
64         return beta;
65     }
66     int bestMoveAB(Board board, Player player) {
67         float bestEval = (float) Integer.MIN_VALUE;
68         float alpha = (float) Integer.MIN_VALUE;
69         float beta = (float) Integer.MAX_VALUE;
70         int bestMove = -1;
71         for (int i = 0; i < WIDTH * WIDTH; i++) {
72             Stone s = new Stone(i, player.getColor());
73             if (board.canPut(s)) {
74                 //board.print();
75                 board.setBoard(s); //BLACKに打たせるときは、-player.getColor()の負
76                 号を取り、
77                 float eval = alphabeta(board, 6, -player.getColor(), player, alpha,
78                     beta);
79                 board.setEmpty(i); //Board.evaluate で -1 と 1 を入れ替える
80                 //System.out.println("k="+i+", eval="+eval);
81                 if (eval > bestEval) {
82                     bestEval = eval;
83                 }
84             }
85         }
86         return bestMove;
87     }
88 }
```

```
79         bestMove = i;
80     }
81 }
82 //System.out.print(i+"\t");
83 }
84 return bestMove;
85 }
86 }
```

ソースコード 11 標準入力：戦略クラス内：N 目並べ

```
1  int prompt(String name) {
2      // 標準入力から
3      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
4      int num = -1;
5      String fmt = null, instr = null;
6      do {
7          fmt = String.format("\n石を置く場所 xy を指定 ( %s の番です ) :", name);
8          System.out.print(fmt);
9          try {
10             instr = br.readLine();
11             char cx = instr.trim().charAt(0);
12             int nx;
13             if (numP(cx)) {
14                 nx = Integer.parseInt(String.valueOf(cx));
15             } else {
16                 nx = fromA(cx) + 10;
17             }
18             char cy = instr.trim().charAt(1);
19             int ny;
20             if (numP(cy)) {
21                 ny = Integer.parseInt(String.valueOf(cy));
22             } else {
23                 ny = fromA(cy) + 10;
24             }
25             num = nx * WIDTH + ny;
26         } catch (Exception e) {
27             e.printStackTrace();
28         }
29         if (!(0 <= num && num < WIDTH * WIDTH)) {
30             System.out.println("やり直し! 場所を xy で指定して ");
31             continue;
32         }
33         break;
34     } while (true);
35     return num;
36 }
```

参考文献

[1]