* 1. Othelloクラス

以下のドライバを用いてOthelloクラスの単体テストを行った。なお、スタブは使用していない。

|  |
| --- |
| import java.io.BufferedReader;  import java.io.InputStreamReader;  public class OthelloDriver {  public static void main (String [] args) throws Exception{  BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 1);  Othello game = new Othello(); //初期化  System.out.println("テスト2：結果：");  System.out.println("start()を行う前のgetTurn()は : "+game.getTurn());  game.start();  System.out.println("start()を行った後のgetTurn()は : "+game.getTurn());  //printStatus(game);  //printGrids(game);  while(true){  printStatus(game);  printGrids(game);  if(game.pass(game.getTurn())==0) {  //System.out.println("手番は " + game.getTurn());  System.out.println("石を置く場所の行をキーボードで入力してください");  String si = r.readLine();//文字列の入力  System.out.println("石を置く場所の列をキーボードで入力してください");  String sj = r.readLine();  //System.out.println("");  int pt = game.put(game.getTurn(), Integer.parseInt(si), Integer.parseInt(sj));  System.out.println("put()出力(0 : 置けた, 1 : 置けなかった) : "+pt);  if(pt==0) {  System.out.printf("\n[%s, %s]に石を置きました.\n", si, sj);  printStatus(game);  printGrids(game);  }else {  System.out.println("置けない場所です. 再入力してください");  }  }  else {  System.out.println("置ける場所がありませんでした。パス.");  game.changeTurn();  }  if(game.check\_end(0)==1)  break;  }    }  //状態を表示する  public static void printStatus(Othello game){  //System.out.println("result出力 (0: 引き分け 1,-1: 枚数での勝敗 2,-2: 時間切れでの勝敗 3,-3: 投了 それ以外:切断など)：" + game.result()); //勝敗の決定  System.out.println("check\_end出力 (0 : 終了していない, 1 : 終了した) : " + game.check\_end(0)); //対局終了の判定  if(game.check\_end(0)==1)  System.out.println("result出力 (終了時:±1, 全てのマスが埋まっていない場合:±2, 引き分け(全埋まり):100, 引き分け（埋まっていない):-100)：" + game.result()); //勝敗の決定  System.out.println("getTurn出力 (1 : 黒, -1 : 白)：" + game.getTurn()); //て番情報の取得  System.out.println("pass出力 (0 : パスしない, 1 : パスする) : "+game.pass(game.getTurn()));    }  //テスト用に盤面を表示する  public static void printGrids(Othello game){  game.print();  }  } |

OthelloDriver.javaの出力結果 :

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  テスト1ではコンストラクタを初期化した上でresultとpass以外のメソッドを一通り実装した上でそれらの正当性を示す。1以降では、これらの単体テストが出来たとみなして着目すべき点である、決着がつく直前のみをペーストしている。  テスト2では黒がパスをするときに白の番がくることを示した上で片方が全滅したときの処理を示している。result = 2(埋まらず黒win)  これ以降では、resultが変わる盤面を作り出し、resultの値を出力する。  3 : result = 100(埋まって引き分け)  4 : result = -100(埋まらず引き分け)  5 : result = 1(埋まって黒win)  6 : result = -2(埋まり切らず白win)  7 : result = -1(埋まって白win)  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  テスト１：Othelloクラスのオブジェクトを初期化した結果：  start()を行う前のgetTurn()は : 100  start()を行った後のgetTurn()は : 1  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：1  pass出力 (0 : パスしない, 1 : パスする) : 0  1 置ける場所:\*  0 1 2 3 4 5 6 7  0| | | | | | | | |  1| | | | | | | | |  2| | | | +| \*| | | |  3| | | +|〇|●| \*| | |  4| | | \*|●|〇| +| | |  5| | | | \*| +| | | |  6| | | | | | | | |  7| | | | | | | | |  石を置く場所の行をキーボードで入力してください  2  石を置く場所の列をキーボードで入力してください  2  put()出力(0 : 置けた, 1 : 置けなかった) : 1  置けない場所です. 再入力してください  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：1  pass出力 (0 : パスしない, 1 : パスする) : 0  1 置ける場所:\*  0 1 2 3 4 5 6 7  0| | | | | | | | |  1| | | | | | | | |  2| | | | +| \*| | | |  3| | | +|〇|●| \*| | |  4| | | \*|●|〇| +| | |  5| | | | \*| +| | | |  6| | | | | | | | |  7| | | | | | | | |  石を置く場所の行をキーボードで入力してください  4  石を置く場所の列をキーボードで入力してください  2  put()出力(0 : 置けた, 1 : 置けなかった) : 0  [4, 2]に石を置きました.  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：-1  pass出力 (0 : パスしない, 1 : パスする) : 0  -1 置ける場所:+  0 1 2 3 4 5 6 7  0| | | | | | | | |  1| | | | | | | | |  2| | | | | \*| \*| | |  3| | | +|〇|●| \*| | |  4| | |〇|〇|〇| | | |  5| | | +| | +| | | |  6| | | | | | | | |  7| | | | | | | | |  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：-1  pass出力 (0 : パスしない, 1 : パスする) : 0  -1 置ける場所:+  0 1 2 3 4 5 6 7  0| | | | | | | | |  1| | | | | | | | |  2| | | | | \*| \*| | |  3| | | +|〇|●| \*| | |  4| | |〇|〇|〇| | | |  5| | | +| | +| | | |  6| | | | | | | | |  7| | | | | | | | |  石を置く場所の行をキーボードで入力してください  5  石を置く場所の列をキーボードで入力してください  3  put()出力(0 : 置けた, 1 : 置けなかった) : 1  置けない場所です. 再入力してください  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：-1  pass出力 (0 : パスしない, 1 : パスする) : 0  -1 置ける場所:+  0 1 2 3 4 5 6 7  0| | | | | | | | |  1| | | | | | | | |  2| | | | | \*| \*| | |  3| | | +|〇|●| \*| | |  4| | |〇|〇|〇| | | |  5| | | +| | +| | | |  6| | | | | | | | |  7| | | | | | | | |  石を置く場所の行をキーボードで入力してください  5  石を置く場所の列をキーボードで入力してください  4  put()出力(0 : 置けた, 1 : 置けなかった) : 0  [5, 4]に石を置きました.  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：1  pass出力 (0 : パスしない, 1 : パスする) : 0  1 置ける場所:\*  0 1 2 3 4 5 6 7  0| | | | | | | | |  1| | | | | | | | |  2| | | +| | | \*| | |  3| | | +|〇|●| \*| | |  4| | +|〇|〇|●| \*| | |  5| | | +| |●| \*| | |  6| | | | | | \*| | |  7| | | | | | | | |  ----------------------------------------------------------------  テスト2：パスの判定とコマの全滅が起こったときの結果：  start()を行う前のgetTurn()は : 100  start()を行った後のgetTurn()は : 1  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：1  pass出力 (0 : パスしない, 1 : パスする) : 1  1 置ける場所:\*  0 1 2 3 4 5 6 7  0| | | | | | | | |  1| | +| | +| | +| | |  2| | |〇|〇|〇| | | |  3| | +|〇|●|〇| +| | |  4| | |〇|〇|〇| | | |  5| | +| | +| | +| | |  6| | | | | | | | |  7| | | | | | | | |  置ける場所がありませんでした。パス.  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：-1  pass出力 (0 : パスしない, 1 : パスする) : 0  -1 置ける場所:+  0 1 2 3 4 5 6 7  0| | | | | | | | |  1| | +| | +| | +| | |  2| | |〇|〇|〇| | | |  3| | +|〇|●|〇| +| | |  4| | |〇|〇|〇| | | |  5| | +| | +| | +| | |  6| | | | | | | | |  7| | | | | | | | |  石を置く場所の行をキーボードで入力してください  1  石を置く場所の列をキーボードで入力してください  1  put()出力(0 : 置けた, 1 : 置けなかった) : 0  [1, 1]に石を置きました.  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：1  pass出力 (0 : パスしない, 1 : パスする) : 0  1 置ける場所:\*  0 1 2 3 4 5 6 7  0| \*| | | | | | | |  1| |●| \*| +| | +| | |  2| | \*|●|〇|〇| +| | |  3| | +|〇|●|〇| +| | |  4| | |〇|〇|〇| | | |  5| | +| +| +| | +| | |  6| | | | | | | | |  7| | | | | | | | |  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：1  pass出力 (0 : パスしない, 1 : パスする) : 0  1 置ける場所:\*  0 1 2 3 4 5 6 7  0| \*| | | | | | | |  1| |●| \*| +| | +| | |  2| | \*|●|〇|〇| +| | |  3| | +|〇|●|〇| +| | |  4| | |〇|〇|〇| | | |  5| | +| +| +| | +| | |  6| | | | | | | | |  7| | | | | | | | |  石を置く場所の行をキーボードで入力してください  0  石を置く場所の列をキーボードで入力してください  0  put()出力(0 : 置けた, 1 : 置けなかった) : 0  [0, 0]に石を置きました.  no one can put  check\_end出力 (0 : 終了していない, 1 : 終了した) : 1  no one can put  result出力 (終了時:±1, 全てのマスが埋まっていない場合:±2, 引き分け(全埋まり):100, 引き分け（埋まっていない):-100)：2  getTurn出力 (1 : 黒, -1 : 白)：-1  pass出力 (0 : パスしない, 1 : パスする) : 1  -1 置ける場所:+  0 1 2 3 4 5 6 7  0|〇| | | | | | | |  1| |〇| | | | | | |  2| | |〇|〇|〇| | | |  3| | |〇|〇|〇| | | |  4| | |〇|〇|〇| | | |  5| | | | | | | | |  6| | | | | | | | |  7| | | | | | | | |  no one can put  --------------------------------------------------------------------  テスト3：すべてのマスが埋まって引き分けだった結果  start()を行う前のgetTurn()は : 100  start()を行った後のgetTurn()は : 1  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：1  pass出力 (0 : パスしない, 1 : パスする) : 0  1 置ける場所:\*  0 1 2 3 4 5 6 7  0|●|●|●|●| \*|〇|〇|〇|  1|●|●|●|●|●|〇|〇|〇|  2|●|●|●|●|〇|〇|〇|〇|  3|●|●|●|●|〇|〇|〇|〇|  4|●|●|●|●|〇|〇|〇|〇|  5|●|●|●|●|〇|〇|〇|〇|  6|●|●|●|●|〇|〇|〇|〇|  7|●|●|●|●|〇|〇|〇|〇|  石を置く場所の行をキーボードで入力してください  0  石を置く場所の列をキーボードで入力してください  4  put()出力(0 : 置けた, 1 : 置けなかった) : 0  [0, 4]に石を置きました.  no one can put  check\_end出力 (0 : 終了していない, 1 : 終了した) : 1  no one can put  result出力 (終了時:±1, 全てのマスが埋まっていない場合:±2, 引き分け(全埋まり):100, 引き分け（埋まっていない):-100)：100  getTurn出力 (1 : 黒, -1 : 白)：-1  pass出力 (0 : パスしない, 1 : パスする) : 1  -1 置ける場所:+  0 1 2 3 4 5 6 7  0|●|●|●|●|〇|〇|〇|〇|  1|●|●|●|●|〇|〇|〇|〇|  2|●|●|●|●|〇|〇|〇|〇|  3|●|●|●|●|〇|〇|〇|〇|  4|●|●|●|●|〇|〇|〇|〇|  5|●|●|●|●|〇|〇|〇|〇|  6|●|●|●|●|〇|〇|〇|〇|  7|●|●|●|●|〇|〇|〇|〇|  no one can put  ---------------------------------------------------------------------  テスト4：マスが埋まっていない状態で引き分けになった結果  start()を行う前のgetTurn()は : 100  start()を行った後のgetTurn()は : 1  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：1  pass出力 (0 : パスしない, 1 : パスする) : 0  1 置ける場所:\*  0 1 2 3 4 5 6 7  0|〇|〇| \*| \*| |●|●|●|  1|〇|〇|●| \*| |●|●|●|  2|〇|〇|〇| \*| |●|●|●|  3|〇|〇|〇| | |●|●|●|  4|〇|〇|〇| | |●|●|●|  5|〇|〇|〇| | |●|●|●|  6|〇|〇|〇| | |●|●|●|  7|〇|〇|〇| | |●|●|●|  石を置く場所の行をキーボードで入力してください  0  石を置く場所の列をキーボードで入力してください  2  put()出力(0 : 置けた, 1 : 置けなかった) : 0  [0, 2]に石を置きました.  no one can put  check\_end出力 (0 : 終了していない, 1 : 終了した) : 1  no one can put  result出力 (終了時:±1, 全てのマスが埋まっていない場合:±2, 引き分け(全埋まり):100, 引き分け（埋まっていない):-100)：-100  getTurn出力 (1 : 黒, -1 : 白)：-1  pass出力 (0 : パスしない, 1 : パスする) : 1  -1 置ける場所:+  0 1 2 3 4 5 6 7  0|〇|〇|〇| | |●|●|●|  1|〇|〇|〇| | |●|●|●|  2|〇|〇|〇| | |●|●|●|  3|〇|〇|〇| | |●|●|●|  4|〇|〇|〇| | |●|●|●|  5|〇|〇|〇| | |●|●|●|  6|〇|〇|〇| | |●|●|●|  7|〇|〇|〇| | |●|●|●|  no one can put  テスト5：すべてのマスが埋まって黒が勝った結果  start()を行う前のgetTurn()は : 100  start()を行った後のgetTurn()は : 1  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：1  pass出力 (0 : パスしない, 1 : パスする) : 0  1 置ける場所:\*  0 1 2 3 4 5 6 7  0| \*|●|〇|〇|〇|〇|〇|〇|  1|〇|〇|〇|〇|〇|〇|〇|〇|  2|〇|〇|〇|〇|〇|〇|〇|〇|  3|〇|〇|〇|〇|〇|〇|〇|〇|  4|〇|〇|〇|〇|〇|〇|〇|〇|  5|〇|〇|〇|〇|〇|〇|〇|〇|  6|〇|〇|〇|〇|〇|〇|〇|〇|  7|〇|●|〇|〇|〇|〇|〇|〇|  石を置く場所の行をキーボードで入力してください  0  石を置く場所の列をキーボードで入力してください  0  put()出力(0 : 置けた, 1 : 置けなかった) : 0  [0, 0]に石を置きました.  no one can put  check\_end出力 (0 : 終了していない, 1 : 終了した) : 1  no one can put  result出力 (終了時:±1, 全てのマスが埋まっていない場合:±2, 引き分け(全埋まり):100, 引き分け（埋まっていない):-100)：1  getTurn出力 (1 : 黒, -1 : 白)：-1  pass出力 (0 : パスしない, 1 : パスする) : 1  -1 置ける場所:+  0 1 2 3 4 5 6 7  0|〇|〇|〇|〇|〇|〇|〇|〇|  1|〇|〇|〇|〇|〇|〇|〇|〇|  2|〇|〇|〇|〇|〇|〇|〇|〇|  3|〇|〇|〇|〇|〇|〇|〇|〇|  4|〇|〇|〇|〇|〇|〇|〇|〇|  5|〇|〇|〇|〇|〇|〇|〇|〇|  6|〇|〇|〇|〇|〇|〇|〇|〇|  7|〇|●|〇|〇|〇|〇|〇|〇|  no one can put  テスト6：マスが埋まらない状態で白が勝った結果 :  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：-1  pass出力 (0 : パスしない, 1 : パスする) : 0  -1 置ける場所:+  0 1 2 3 4 5 6 7  0| |〇|〇| | +| +|●|●|  1|〇|〇|〇| | +|〇|●|●|  2|〇|〇|〇| | +|〇|●|●|  3|〇|〇|〇| | +|〇|●|●|  4|〇|〇|〇| | +|●|●|●|  5|〇|〇|〇| | |●|●|●|  6|〇|〇|〇| | |●|●|●|  7|〇|〇|〇| | |●|●|●|  石を置く場所の行をキーボードで入力してください  0  石を置く場所の列をキーボードで入力してください  5  put()出力(0 : 置けた, 1 : 置けなかった) : 0  [0, 5]に石を置きました.  no one can put  check\_end出力 (0 : 終了していない, 1 : 終了した) : 1  no one can put  result出力 (終了時:±1, 全てのマスが埋まっていない場合:±2, 引き分け(全埋まり):100, 引き分け（埋まっていない):-100)：-2  getTurn出力 (1 : 黒, -1 : 白)：1  pass出力 (0 : パスしない, 1 : パスする) : 1  1 置ける場所:\*  0 1 2 3 4 5 6 7  0| |〇|〇| | |●|●|●|  1|〇|〇|〇| | |●|●|●|  2|〇|〇|〇| | |●|●|●|  3|〇|〇|〇| | |●|●|●|  4|〇|〇|〇| | |●|●|●|  5|〇|〇|〇| | |●|●|●|  6|〇|〇|〇| | |●|●|●|  7|〇|〇|〇| | |●|●|●|  no one can put  テスト7：すべてのマスが埋まって白が勝った結果：  check\_end出力 (0 : 終了していない, 1 : 終了した) : 0  getTurn出力 (1 : 黒, -1 : 白)：-1  pass出力 (0 : パスしない, 1 : パスする) : 0  -1 置ける場所:+  0 1 2 3 4 5 6 7  0|〇|●|●|●|●| !|●|●|  1|〇|〇|〇|〇|〇|〇|●|●|  2|〇|●|●|●|〇|〇|●|●|  3|〇|●|●|〇|●|〇|●|●|  4|〇|●|〇|●|●|●|●|●|  5|〇|〇|●|●|●|●|●|●|  6|〇|●|●|●|●|●|●|●|  7|〇|●|●|●|●|●|●|●|  石を置く場所の行をキーボードで入力してください  0  石を置く場所の列をキーボードで入力してください  5  put()出力(0 : 置けた, 1 : 置けなかった) : 0  [0, 5]に石を置きました.  no one can put  check\_end出力 (0 : 終了していない, 1 : 終了した) : 1  no one can put  result出力 (終了時:±1, 全てのマスが埋まっていない場合:±2, 引き分け(全埋まり):100, 引き分け（埋まっていない):-100)：-1  getTurn出力 (1 : 黒, -1 : 白)：1  pass出力 (0 : パスしない, 1 : パスする) : 1  1 置ける場所:\*  0 1 2 3 4 5 6 7  0|〇|●|●|●|●|●|●|●|  1|〇|〇|〇|〇|●|●|●|●|  2|〇|●|●|●|〇|●|●|●|  3|〇|●|●|〇|●|●|●|●|  4|〇|●|〇|●|●|●|●|●|  5|〇|〇|●|●|●|●|●|●|  6|〇|●|●|●|●|●|●|●|  7|〇|●|●|●|●|●|●|●|  no one can put |

(ウ) ClientDriver.java

以下のドライバを用いてClientクラスの単体テストを行った。なお、スタブは使用していない。

|  |
| --- |
| import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  public class ClientDriver extends Exception{  private Client oclient;  int puttern = 0;  ClientDriver(){  BufferedReader r = new BufferedReader(new InputStreamReader(System.in), 1);  oclient = new Client();  System.out.println("名前を入力してください");  oclient.prepare();//名前の準備、エコーサーバに接続  oclient.matching();//マッチング  String st = null;  System.out.println("お名前が入力されました。");  System.out.println("色を決めるため、bを入力してください");//wは未実装  try {  st = r.readLine();  } catch (IOException e) {  e.printStackTrace();  }  oclient.sendMessage(st);//st=bを送って黒側として接続。  while(true) {  try {  Thread.sleep(1000);  } catch (InterruptedException e) {  e.printStackTrace();  }  if(oclient.getStatus()==1||oclient.getStatus()==-1) {  oclient.delete();//画面の初期化  break;  }  }  System.out.println("あなたの色は黒です。次にあなたの制限時間を決めます...");  if(oclient.getStatus()==1) {//こちら側が600秒を申請する。  oclient.setLimitTime();//普通に送信すると、まだ宣言していない  oclient.setInputTime(600);//acceptが呼び出されるので、疑似的に送信しているように見せかけている  while(true) {  try {  Thread.sleep(10000);  } catch (InterruptedException e) {  e.printStackTrace();  }  oclient.sendMessage("AAccept");  try {  Thread.sleep(1000);  } catch (InterruptedException e) {  e.printStackTrace();  }  if(String.valueOf(oclient.getdecided())=="true") {  oclient.resetselected();  oclient.resetdecided();  oclient.delete();  System.out.println("承認されました。あなたの制限時間は600秒です。");  break;  }  }  oclient.stopper\_input();  oclient.accept();  try {  Thread.sleep(3000);  } catch (InterruptedException e) {  e.printStackTrace();  }  while(true) {  try {  if(puttern == 0) {  oclient.sendMessage("A300");  oclient.setInputTime(300);  System.out.println("相手が300秒を要求します。「多くしろ」と拒否してください");  Thread.sleep(10000);  System.out.println("拒否完了");  }  else if(puttern == 1){  Thread.sleep(1000);  oclient.sendMessage("A900");  oclient.setInputTime(900);  System.out.println("相手が900秒を要求します。「少なくしろ」と拒否してください");  Thread.sleep(10000);  System.out.println("拒否完了");  }  else if(puttern == 2){  oclient.sendMessage("A750");  oclient.setInputTime(750);  System.out.println("相手が750秒を要求します。「少なくしろ」と拒否してください");  Thread.sleep(10000);  System.out.println("拒否完了");  }  else {  oclient.sendMessage("A600");  oclient.setInputTime(600);  System.out.println("相手が600秒を要求します。承認してください");  oclient.setvalue();  Thread.sleep(10000);  System.out.println("承認完了。相手の制限時間は600秒です。");  }  } catch (InterruptedException e) {  // TODO 自動生成された catch ブロック  e.printStackTrace();  }  oclient.setvalue();  if(puttern == 0)  puttern = 1;  else if(puttern == 1)  puttern = 2;  else if(puttern == 2)  puttern = 3;  else  break;  }  }  oclient.delete();  System.out.println("置いたときの操作を1ターンずつ行います。黒いコマを置いてください : ");  oclient.Playing();  while(true) {  try {  System.out.println();  Thread.sleep(10000);  } catch (InterruptedException e) {  e.printStackTrace();  }  if(oclient.changed()==-1) {  //oclient.delete();  //oclient.updateDisp("800");  System.out.println("あなたの色を白にしました.白を置いてください : ");  oclient.changeTurn();  break;  }  }  while(true) {  try {  System.out.println();  Thread.sleep(10000);  } catch (InterruptedException e) {  e.printStackTrace();  }  if(oclient.changed()==1) {  System.out.println("投了ボタンを押してください.");  break;  }  }  }  public static void main(String args[]) {  ClientDriver cd = new ClientDriver();  }  } |

ClientDriver.javaの出力結果

テキスト

自動的に生成された説明

出力結果(UI) :

|  |  |  |
| --- | --- | --- |
| グラフィカル ユーザー インターフェイス, アプリケーション  自動的に生成された説明1 |  |  |
| テキスト, 手紙  自動的に生成された説明2 | グラフィカル ユーザー インターフェイス, アプリケーション, Word  自動的に生成された説明3 | グラフィカル ユーザー インターフェイス, アプリケーション  自動的に生成された説明4 |
| グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール  自動的に生成された説明5 | グラフィカル ユーザー インターフェイス, アプリケーション  自動的に生成された説明6 | グラフィカル ユーザー インターフェイス, アプリケーション  自動的に生成された説明7 |

|  |  |  |
| --- | --- | --- |
| グラフ が含まれている画像  自動的に生成された説明8 | グラフ  低い精度で自動的に生成された説明9 | グラフ が含まれている画像  自動的に生成された説明10 |