

# 試験報告書

オセロゲーム開発

G グループ

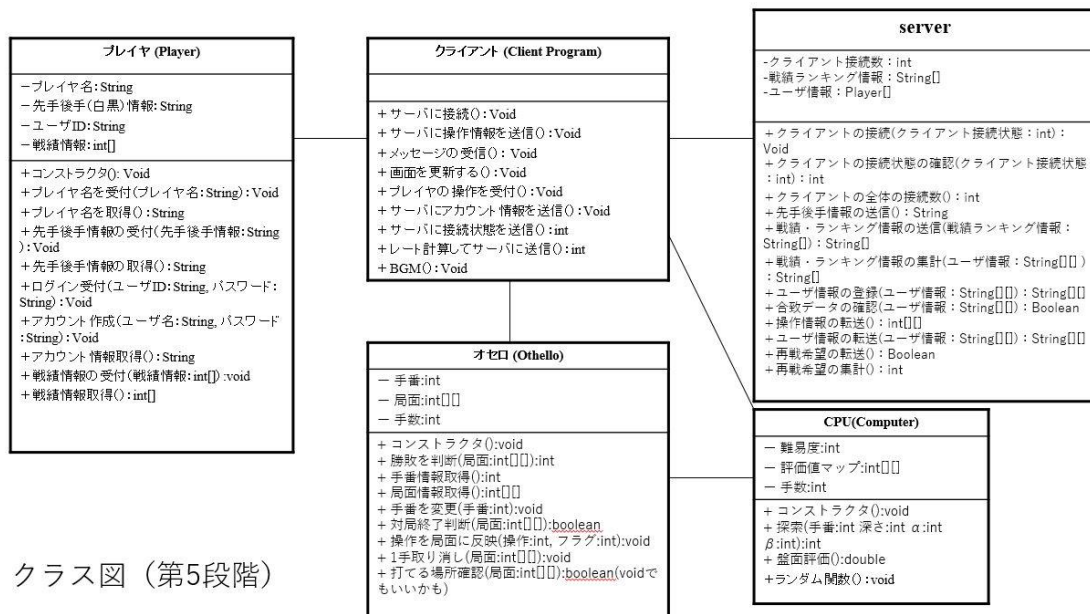
提出日：2020 年 8 月 14 日

## 1. はじめに

本報告書は、ネットワーク対戦型オセロゲームシステムの試験方法および試験結果を記すものである。試験対象であるシステムのソースコードおよびクラス図は以下の通りである。

ソースコード一覧：

Player.java	PlayerDriver.java (テスト用ドライバ)
Client.java	ClientDriver.java (テスト用ドライバ)
Othello.java	OthelloDriver.java(テスト用ドライバ)
Server.java	ServerDriver.java(テスト用ドライバ)
Computer.java	ComputerDriver.java(テスト用ドライバ)

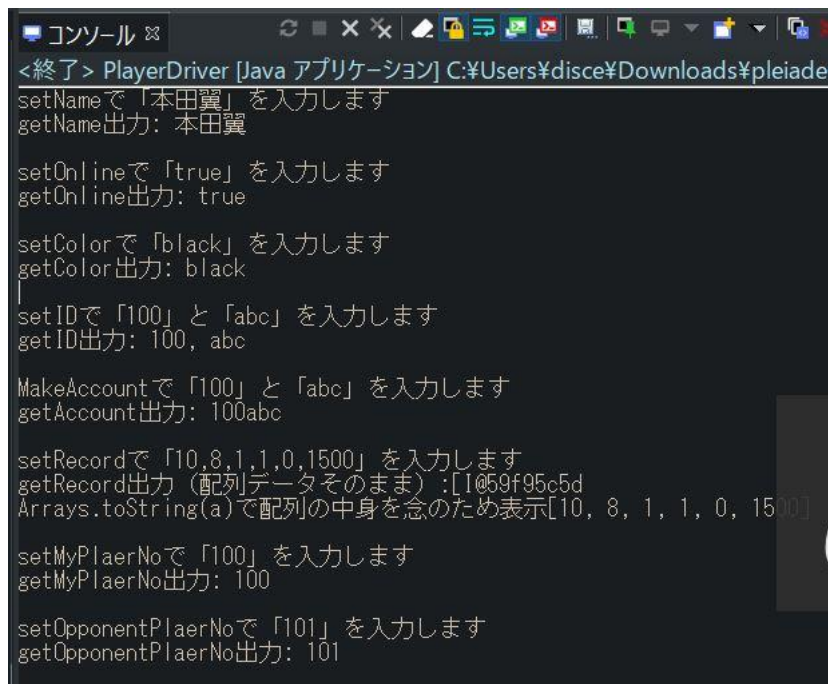


## 単体テスト

### (ア) Player クラス

試験報告書に添付する PlayerDriver.java を用いて Player クラスの単体テストを行った。

試験結果は以下の通り



```
<終了> PlayerDriver [Java アプリケーション] C:\Users\disce\Downloads\pleiade
setNameで「本田翼」を入力します
getName出力: 本田翼

setOnlineで「true」を入力します
getOnline出力: true

setColorで「black」を入力します
getColor出力: black

setIDで「100」と「abc」を入力します
getID出力: 100, abc

MakeAccountで「100」と「abc」を入力します
getAccount出力: 100abc

setRecordで「10,8,1,1,0,1500」を入力します
getRecord出力(配列データそのまま): [1059f95c5d
Arrays.toString(a)で配列の中身を念のため表示[10, 8, 1, 1, 0, 1500]

setMyPlaerNoで「100」を入力します
getMyPlaerNo出力: 100

setOpponentPlaerNoで「101」を入力します
getOpponentPlaerNo出力: 101
```

### (イ) Othello クラス

試験報告書に添付する OthelloDriver.java 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。

OthelloDriver.java は以下の内容を入力する。

通常ルール・スペシャルルールそれぞれにおいて

- checkWinner 出力(試合が終了している場合にどちらの色が勝っているかを表示)
- getWhitestone/getBlackstone 出力(黒石、白石のを出力)
- isGameover 出力(対局が終了しているかどうかを出力)
- getTurn 出力(どちらの手番かを出力: 白は 1/黒は-1 となっている)

- grids 出力(盤面を表現する二次元配列に格納される整数を図形に変換して出力)
- s\_setStone 出力(イベントの効果を示す整数が出力される。\* スペシャルルールの場合のみ表示)
- スペシャルルールの場合のイベントを実行して盤面を出力

試験結果は以下の通り

○通常ルールの場合

```

コンソール
OthelloDriver [Java アプリケーション] C:\Users\Krymt\Downloads\pleiades\java\11\bin\javaw.exe (2020/08/07 15:37:59)
通常ルールなら1を入力・特殊ルールなら2を入力
mode=1
テスト 1 : Othelloクラスのオブジェクトを初期化した結果 :
checkWinner出力:not_finished
getWhitestone出力:2
getBlackstone出力:2
isGameOver出力:false
getTurn出力:
-1(black)getRow出力:8
Gridsテスト出力(8要素ごとに改行,△はおける場所の表示です。)
 0 1 2 3 4 5 6 7
0 □ □ □ □ □ □ □
1 □ □ □ □ □ □ □
2 □ □ □ △ □ □ □
3 □ □ △ ○ ● □ □
4 □ □ ● ○ △ □ □
5 □ □ □ △ □ □ □
6 □ □ □ □ □ □ □
7 □ □ □ □ □ □ □
石を置く場所(数字)をキーボードで入力してください
x=|

石を置く場所(数字)をキーボードで入力してください
x=3
y=2
k=3y=3 が入力されました。手番は -1 です。(-1=black,l=white)
手番を変更します。
checkWinner出力:not_finished
getWhitestone出力:1
getBlackstone出力:4
isGameOver出力:false
getTurn出力:
1(white)getRow出力:8
Gridsテスト出力(8要素ごとに改行,△はおける場所の表示です。)
 0 1 2 3 4 5 6 7
0 □ □ □ □ □ □ □
1 □ □ □ □ □ □ □
2 □ □ △ ● △ □ □
3 □ □ ● ● □ □ □
4 □ □ △ ● ○ □ □
5 □ □ □ □ □ □ □
6 □ □ □ □ □ □ □
7 □ □ □ □ □ □ □
石を置く場所(数字)をキーボードで入力してください
x=

```

```

コンソール
OthelloDriver [Java アプリケーション] C:\Users\Krymt\Downloads\pleiades\ja
isGameOver出力:false
getTurn出力:
-1(black)getRow出力:8
Gridsテスト出力(8要素ごとに改行,△はおける場所の表示です。)
  0 1 2 3 4 5 6 7
0 □ □ △ △ △ □ □ □
1 □ □ △ ○ □ □ □ □
2 □ ● ● ○ ● □ □ □
3 □ □ △ ○ ● □ □ □
4 □ □ △ ● ● □ □ □
5 □ □ □ □ □ □ □ □
6 □ □ □ □ □ □ □ □
7 □ □ □ □ □ □ □ □
石を置く場所(数字)をキーボードで入力してください
x=3
y=0
k=3y=3 が入力されました。手番は -1 です。(-1=black,1=white)
手番を変更します。
checkWinner出力:black win
getWhitestone出力:0
getBlackstone出力:13
isGameOver出力:true
getTurn出力:
1(white)getRow出力:8
Gridsテスト出力(8要素ごとに改行,△はおける場所の表示です。)
  0 1 2 3 4 5 6 7
0 □ □ □ ● □ □ □ □
1 □ □ □ ● □ □ □ □
2 □ ● ● ● ● □ □ □
3 □ □ □ ● ● □ □ □
4 □ □ □ ● ● □ □ □
5 □ □ □ □ □ □ □ □
6 □ □ □ □ □ □ □ □
7 □ □ □ □ □ □ □ □

```

## ○スペシャルルールの場合

- ・初期盤面

```

コンソール
OthelloDriver [Java アプリケーション] C:\Users\Krymt\Downloads\pleiades\java\11\bin\javaw.exe (2020/08/0
通常ルールなら1を入力・特殊ルールなら2を入力
mode=2
テスト 1 : Othelloクラスのオブジェクトを特殊ルールで初期化した結果 :
checkWinner出力:not_finished
getWhitestone出力:2
getBlackstone出力:2
isGameOver出力:false
getTurn出力:
-1(black)getRow出力:8
Gridsテスト出力(8要素ごとに改行△はおける場所の表示。×は設置不可、☆はイベントマスです)
  0 1 2 3 4 5 6 7
0 □ □ □ □ □ □ □ □
1 □ ☆ □ □ □ □ ☆ □
2 □ □ □ △ □ □ □ □
3 □ ☆ △ ○ ● □ □ □
4 ☆ □ □ ● ○ △ □ □
5 □ ☆ □ □ △ □ □ ☆
6 □ □ □ □ ☆ □ □ □
7 □ □ □ □ □ □ □ □

```

- ・暗闇イベント (s\_setStone 出力のみ)

Gridsテスト出力(8要素ごとに改行△はおける場所の表示。×は設置不可、☆はイベントマスです)

```
  0 1 2 3 4 5 6 7
0 □ □ □ □ □ □ □
1 □ ☆ △ □ □ □ ☆ □
2 □ △ ○ ○ ● □ □ □
3 □ △ ○ ○ ● □ □ □
4 △ ○ ● ○ ○ △ □ □
5 □ ● △ □ △ □ ☆
6 □ □ □ □ ☆ ○ □
7 □ □ □ □ □ □ □
```

石を置く場所(数字)をキーボードで入力してください

x=0

y=4

k=0 y=0 が入力されました。手番は -1 です。(-1=black,1=white)

s\_setStone出力7

← ← ← ← ← ← ←

## ・2回行動

↑ ↑ ↑ ↑ ↑ ↑ ↑

checkWinner出力:not\_finished

getWhitestone出力:15

getBlackstone出力:12

isGameOver出力:false

getTurn出力:

-1(black)getRow出力:8

Gridsテスト出力(8要素ごとに改行△はおける場所の表示。×は設置不可、☆はイベントマスです)

```
  0 1 2 3 4 5 6 7
0 □ □ □ □ △ ○ ●
1 □ ☆ □ □ △ ○ ●
2 □ □ ● ● ● ○ ●
3 □ ☆ ○ ● ● ○ ○
4 ○ ○ ○ ○ ○ ○ △
5 △ ○ △ △ ● △ △
6 △ □ □ □ ☆ ● □
7 □ □ □ □ □ □ □
```

石を置く場所(数字)をキーボードで入力してください

x=7

y=5

k=7 y=7 が入力されました。手番は -1 です。(-1=black,1=white)

s\_setStone出力3

```
  0 1 2 3 4 5 6 7
0 □ □ □ □ △ ○ ●
1 □ ☆ □ □ △ ○ ●
2 □ □ ● ● ● ○ ●
3 □ ☆ ○ ● ● ○ ○
4 ○ ○ ○ ○ ○ ● △
5 △ ○ △ △ ● △ ●
6 △ □ □ □ ☆ ● □
7 □ □ □ □ □ □ □
```

もう一度行動できます。石を置く場所を選択してください。

x=

## ・盤面反転

```

OthelloDriver [Java アプリケーション] C:\Users\Krymt\Downloads\pleiades\java\11\bin\javaw.exe (2020/08/07 1
-1(black)getRow出力:8
Gridsテスト出力(8要素ごとに改行△はおける場所の表示。×は設置不可、☆はイベントマスです)
  0 1 2 3 4 5 6 7
0 □ □ □ □ ● □ □ □
1 □ △ △ ● △ △ △ □
2 □ □ ○ ○ ○ ○ □
3 □ △ □ ● ● △ □
4 ☆ □ □ ● □ □ □
5 □ ☆ □ □ ● □ □ ☆
6 □ □ □ □ ☆ □ □
7 □ □ □ □ □ □ □
石を置く場所(数字)をキーボードで入力してください
x=6
y=1
x=8y=6 が入力されました。手番は -1 です。(-1=black,1=white)
s.setStone出力:8
盤面の石の色を全て反転します
手番を変更します。
checkWinner出力:not_finished
getWhitestone出力:9
getBlackstone出力:4
isGameOver出力:false
getTurn出力:
1(white)getRow出力:8
Gridsテスト出力(8要素ごとに改行△はおける場所の表示。×は設置不可、☆はイベントマスです)
  0 1 2 3 4 5 6 7
0 □ □ □ ○ □ □ □
1 □ △ △ ○ △ △ ○ □
2 □ △ ● ● ● ○ △
3 □ △ □ ○ ○ △ □
4 ☆ □ □ ○ □ □ □
5 □ ☆ □ ○ □ □ ☆
6 □ □ □ □ ☆ □ □
7 □ □ □ □ □ □ □
石を置く場所(数字)をキーボードで入力してください
x=
```

・お邪魔マス設置

```

s.setStone出力:8
  0 1 2 3 4 5 6 7
0 □ □ ● ○ △ △ □
1 □ △ □ ● □ ○ △
2 △ ○ ● ○ ○ ○ ○
3 □ ● △ ○ ○ ● □
4 □ □ ○ ○ □ □
5 □ □ △ ○ △ □
6 □ □ □ □ □ □
7 □ □ □ □ □ □
どの色も設置不可となる石を置いてください(空マスに限る)。
x=7
y=7
仲の値3
手番を変更します。
checkWinner出力:not_finished
getWhitestone出力:14
getBlackstone出力:5
isGameOver出力:false
getTurn出力:
1(white)getRow出力:8
Gridsテスト出力(8要素ごとに改行△はおける場所の表示。×は設置不可、☆はイベントマスです)
  0 1 2 3 4 5 6 7
0 □ △ ● ○ □ □ □
1 □ △ □ ● □ ○ □
2 □ ○ ● ○ ○ ○ ○
3 □ ● □ ○ ○ ● △
4 □ △ □ ○ ○ △ □
5 □ □ □ ○ □ □
6 □ □ □ □ □ □
7 □ □ □ □ □ ×
石を置く場所(数字)をキーボードで入力してください
x=
```

・石破壊

```

s_setStone出力1
 0 1 2 3 4 5 6 7
0 □ □ □ □ □ □ □
1 □ ☆ □ □ □ □ ☆
2 □ □ ○ ● □ □ □
3 □ ☆ ○ ● □ □ □
4 ☆ □ ● ○ □ □ □
5 □ ● □ □ □ □ ☆
6 □ □ □ □ ☆ □ □
7 □ □ □ □ □ □ □
破壊する石を選択してください。
x=3
y=2
手番を変更します。
checkWinner出力:not_finished
getWhitestone出力:4
getBlackstone出力:4
isGameOver出力:false
getTurn出力:
1(white)getRow出力:8
Gridsテスト出力(8要素ごとに改行△はおける場所の表示。×は設置不可、☆はイベントマスです)
 0 1 2 3 4 5 6 7
0 □ □ □ □ □ □ □
1 □ ☆ □ □ □ □ ☆
2 □ □ ○ △ △ △ □
3 □ ☆ ○ ● ● □ □
4 ☆ △ ● ○ □ □ □
5 □ ● △ □ □ □ ☆
6 □ □ □ □ ☆ □ □
7 □ □ □ □ □ □ □
石を置く場所(数字)をキーボードで入力してください
x=

```

#### ・ 設置マスを中心に十字方向に石を設置

```

Gridsテスト出力(8要素ごとに改行△はおける場所の表示。×は設置不可、☆はイベントマスです)
 0 1 2 3 4 5 6 7
0 □ □ □ □ □ ○ ● |
1 □ △ △ △ △ ○ ●
2 □ △ ● ● ● ● ●
3 □ △ ● ● ● ○ ○
4 ○ ○ ○ ○ ○ ● △
5 ○ ○ □ □ ● △ ●
6 □ □ □ △ △ ● □
7 □ □ □ □ □ □ △
石を置く場所(数字)をキーボードで入力してください
x=5
y=8
x=5y=5 が入力されました。手番は 1 です。(-1=black,1=white)
s_setStone出力5
上下1マスに石を設置します。
 0 1 2 3 4 5 6 7
0 □ □ □ □ ○ ●
1 □ ☆ □ □ ○ ●
2 □ □ ● ● ○ ●
3 □ ☆ ○ ● ○ ○
4 ○ ○ ○ ○ ● □
5 □ □ □ □ ●
6 ○ ○ ○ ○ ○
7 □ □ □ □ □
手番を変更します。

```

#### ・ 革命（勝利条件が石の少ない方になる）

```

x=1y=1 が入力されました。手番は 1 です。(-1=black,1=white)
s_setStone出力0
手番を変更します。
checkWinner出力:White win
getWhitestone出力:15
getBlackstone出力:48
isGameOver出力:true
getTurn出力:
-1(black)getRow出力:8
Gridsテスト出力(8要素ごとに改行△はおける場所の表示。×は設置不可、☆はイベントマスです)
 0 1 2 3 4 5 6 7
0 ○ ○ ● ● ● ● ●
1 ○ ○ ● ● ● ● ●
2 ○ ○ ● ● ● ● ●
3 ● ○ ● ● ● ● ●
4 ● ○ ● ● ● ● ●
5 ○ × ● ● ● ● ●
6 ○ ● ● ● ● ● ●
7 ● ○ ● ● ● ● ●

```

## (ウ) Client クラス

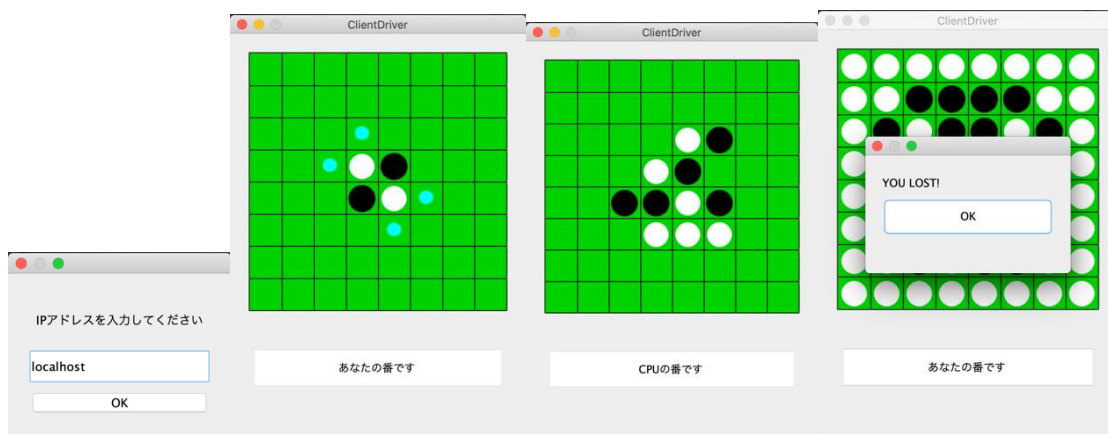
### Client クラス

試験報告書に添付する ClientDriver.java、Othello.java、Computer.java、GreenFrame.jpg、Black.jpg、White.jpg、Placeable.jpg を用いて Client クラスの単体テストを行った。なお、スタブとして ServerDriver.java を使用した。

以下を試験内容とした。

- ・ IP アドレス入力によるサーバへの接続
- ・ オセロゲームの進行
- ・ サーバとの操作情報送受信

試験結果は以下の通り



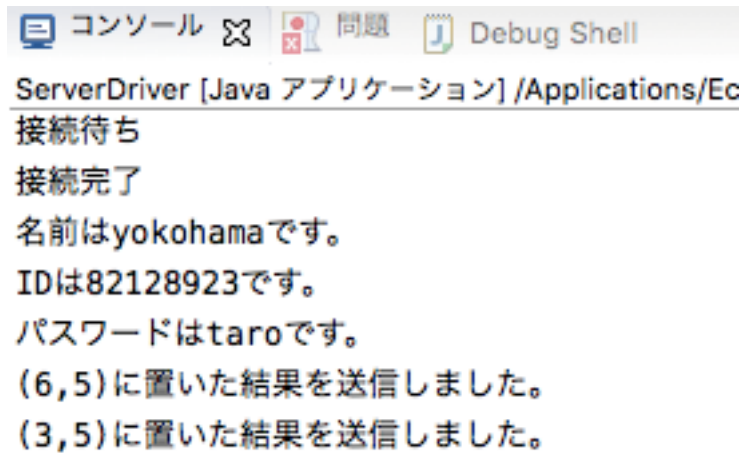
## (エ) Server クラス

試験報告書に添付する ServerDriver.java を用いて Server クラスの単体テストを行った。なお、ClientDriver.java を使用した。

ServerDriver.java は HashMap によるユーザ情報の管理、操作情報の転送を出力する。



試験結果は以下の通り



```
コンソール 問題 Debug Shell
ServerDriver [Java アプリケーション] /Applications/Ec
接続待ち
接続完了
名前はyokohamaです。
IDは82128923です。
パスワードはtaroです。
(6,5)に置いた結果を送信しました。
(3,5)に置いた結果を送信しました。
```

#### (オ) Computer クラス

##### Computer クラス

試験報告書に添付する ComputerDriver.java を用いて Computer 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。

試験結果は以下の通り

---TEST GAME---

01234567  
0|\_|\_|\_|\_|\_|\_|  
1|\_|\_|\_|\_|\_|\_|  
2|\_|\_|\_|\_|\_|\_|  
3|\_|\_|\_|●|\_|\_|  
4|\_|\_|\_|●|\_|\_|  
5|\_|\_|\_|\_|\_|\_|  
6|\_|\_|\_|\_|\_|\_|  
7|\_|\_|\_|\_|\_|\_|

Random:45 → X = 5, Y = 4

01234567  
0|\_|\_|\_|\_|\_|\_|  
1|\_|\_|\_|\_|\_|\_|  
2|\_|\_|\_|\_|\_|\_|  
3|\_|\_|\_|\_|\_|\_|  
4|\_|\_|\_|●|\_|\_|  
5|\_|\_|\_|\_|\_|\_|  
6|\_|\_|\_|\_|\_|\_|  
7|\_|\_|\_|\_|\_|\_|

α-β:35 → X = 5, Y = 3

01234567  
0|\_|\_|\_|\_|\_|\_|  
1|\_|\_|\_|\_|\_|\_|  
2|\_|\_|\_|\_|\_|\_|  
3|\_|\_|\_|\_|\_|\_|  
4|\_|\_|\_|\_|\_|\_|  
5|\_|\_|\_|\_|\_|\_|  
6|\_|\_|\_|\_|\_|\_|  
7|\_|\_|\_|\_|\_|\_|

Random:26 → X = 6, Y = 2

01234567  
0|\_|\_|\_|\_|\_|\_|  
1|\_|\_|\_|\_|\_|\_|  
2|\_|\_|\_|\_|\_|\_|  
3|\_|\_|\_|\_|\_|\_|  
4|\_|\_|\_|\_|\_|\_|  
5|\_|\_|\_|\_|\_|\_|  
6|\_|\_|\_|\_|\_|\_|  
7|\_|\_|\_|\_|\_|\_|

:

01234567  
0|\_|\_|\_|\_|\_|\_|  
1|\_|\_|\_|\_|\_|\_|  
2|\_|\_|\_|\_|\_|\_|  
3|\_|\_|\_|\_|\_|\_|  
4|\_|\_|\_|\_|\_|\_|  
5|\_|\_|\_|\_|\_|\_|  
6|\_|\_|\_|\_|\_|\_|  
7|\_|\_|\_|\_|\_|\_|

Random:67 → X = 7, Y = 6

01234567  
0|\_|\_|\_|\_|\_|\_|  
1|\_|\_|\_|\_|\_|\_|  
2|\_|\_|\_|\_|\_|\_|  
3|\_|\_|\_|\_|\_|\_|  
4|\_|\_|\_|\_|\_|\_|  
5|\_|\_|\_|\_|\_|\_|  
6|\_|\_|\_|\_|\_|\_|  
7|\_|\_|\_|\_|\_|\_|

α-β:75 → X = 5, Y = 7

01234567  
0|\_|\_|\_|\_|\_|\_|  
1|\_|\_|\_|\_|\_|\_|  
2|\_|\_|\_|\_|\_|\_|  
3|\_|\_|\_|\_|\_|\_|  
4|\_|\_|\_|\_|\_|\_|  
5|\_|\_|\_|\_|\_|\_|  
6|\_|\_|\_|\_|\_|\_|  
7|\_|\_|\_|\_|\_|\_|

α-β:77 → X = 7, Y = 7

01234567  
0|\_|\_|\_|\_|\_|\_|  
1|\_|\_|\_|\_|\_|\_|  
2|\_|\_|\_|\_|\_|\_|  
3|\_|\_|\_|\_|\_|\_|  
4|\_|\_|\_|\_|\_|\_|  
5|\_|\_|\_|\_|\_|\_|  
6|\_|\_|\_|\_|\_|\_|  
7|\_|\_|\_|\_|\_|\_|

White win  
white:60black:4

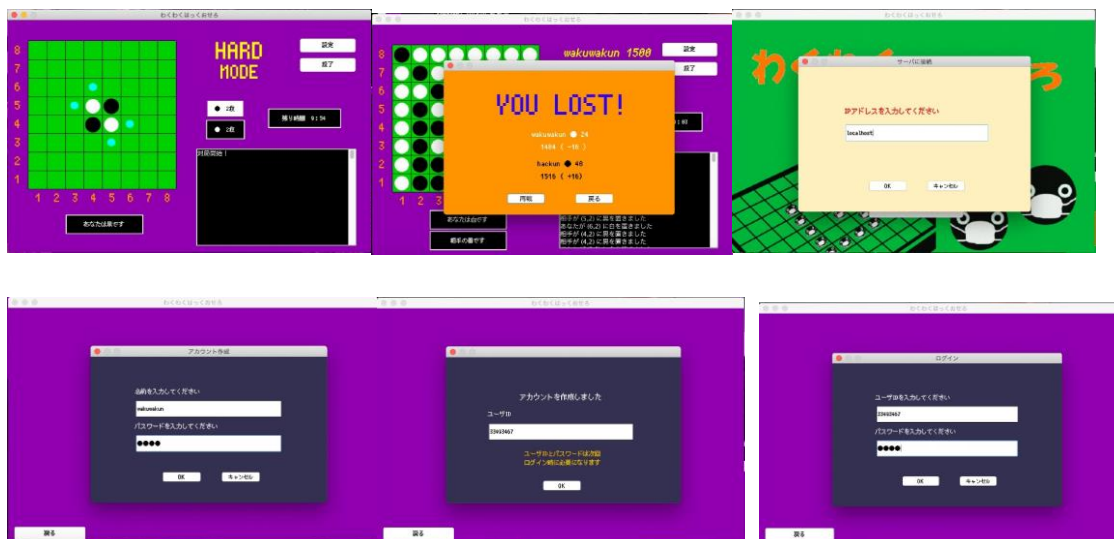
## 2. 結合テスト

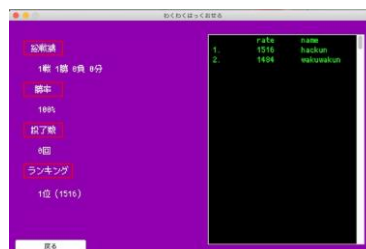
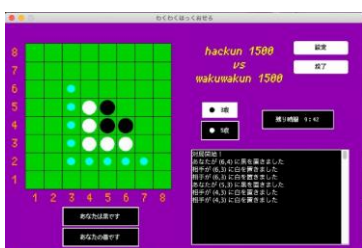
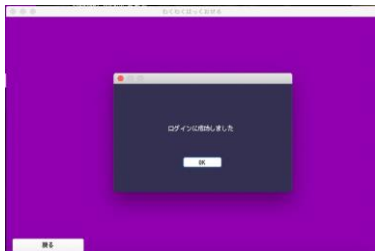
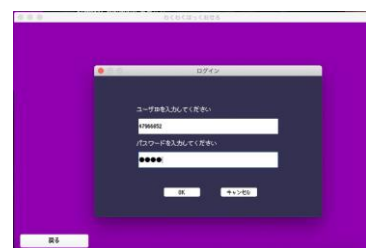
以下の手順に従い、結合テストを行った。

1. Server プログラムを起動
2. Client プログラム 1 を起動し、ローカル対局を選択し、CPU との対戦を進行
3. Client プログラム 1において、ネットワーク対局を押し、IP アドレスを入力
4. アカウント作成を押し、プレイヤ名「hackun」と任意のパスワードを入力
5. Client プログラム 2を起動し、同様にアカウント作成からプレイヤ名「wakuwakun」と任意のパスワードを入力
6. Client プログラム 1でログアウトした後、ユーザ ID とパスワードを入力しログイン
7. Client プログラム 2 でログアウトした後、ユーザ ID とパスワードを入力しログイン
8. ランクマッチを選択し、対局を進行
9. 設定の変更
10. 対局終了後スペシャルマッチを進行し、投了ボタンを押す
11. 再度スペシャルマッチを選択し、対局を進行
12. ゲーム終了後、Client プログラム 1を終了し、サーバから切断
13. ゲーム終了後、Client プログラム 2を終了し、サーバから切断

試験結果を以下に示す。

クライアント側：







サーバ側：

サーバが起動しました。  
 プレイヤ0と接続しました。  
 プレイヤ1と接続しました。  
 33493467 hackun:接続中  
 47966852 wakuwakun:接続中  
 33493467 hackun:接続中  
 プレイヤ 0との接続が切れました。  
 47966852 wakuwakun:接続中  
 33493467 hackun:接続なし  
 プレイヤ2と接続しました。  
 プレイヤ 1との接続が切れました。  
 47966852 wakuwakun:接続なし  
 33493467 hackun:接続なし  
 プレイヤ3と接続しました。  
 47966852 wakuwakun:接続なし  
 33493467 hackun:接続中  
 47966852 wakuwakun:接続中  
 33493467 hackun:接続中  
 プレイヤ 3との接続が切れました。  
 47966852 wakuwakun:接続なし  
 33493467 hackun:接続中  
 プレイヤ 2との接続が切れました。  
 47966852 wakuwakun:接続なし  
 33493467 hackun:接続なし