Othello Project Final Design

Red 14

Shun Nishijima s2977923
Mateusz Bartnicki s3006891

Git-repository
https://gitlab.utwente.nl/software-systems/2022-2023/student-projects/red/red-14

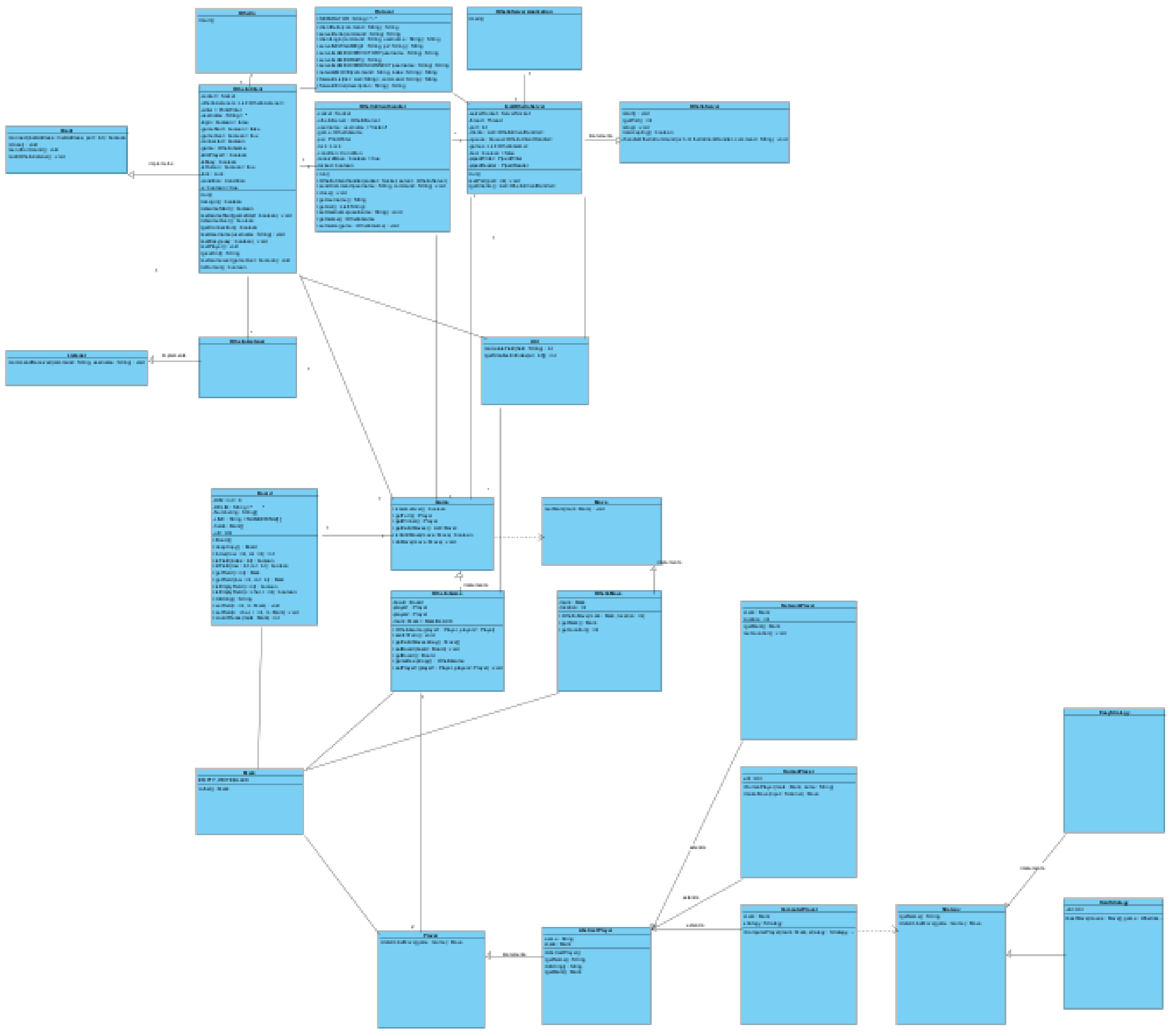
Explanation of realised design

Functional Requirements

1. A standard game can be played on both client and server in conjunction with the reference server and client, respectively.
2. The client can play (on a server) as a human player, controlled by the user.
3. The client can play (on a server) as a computer player, controlled by AI.
4. When the server is started, it will ask the user to input a port number where it will accept connections. If this number is already in use, the server will ask again.
5. When the client is started, it should ask the user for the IP-address and port number of the server to connect to.
6. When the client is controlled by a human player, the user can request a possible valid move as a hint via the TUI.
7. The client can play a full game automatically as the AI without intervention by the user.
8. The user can adjust the AI difficulty via the TUI.
9. Whenever a game has finished (except when the server is disconnected), a new game can be played without needing to establish a new connection in between.
10. All communication outside of playing a game, in particular the handshake and feature negotiation, works on both client and server in conjunction with the reference server and client, respectively.
11. Whenever a client loses connection to a server, the client should gracefully terminate.
12. Whenever Client Disconnects During Game,the server should inform the other client(s) and end the game, allowing the other player to start a new game.

Class Diagram

Overall

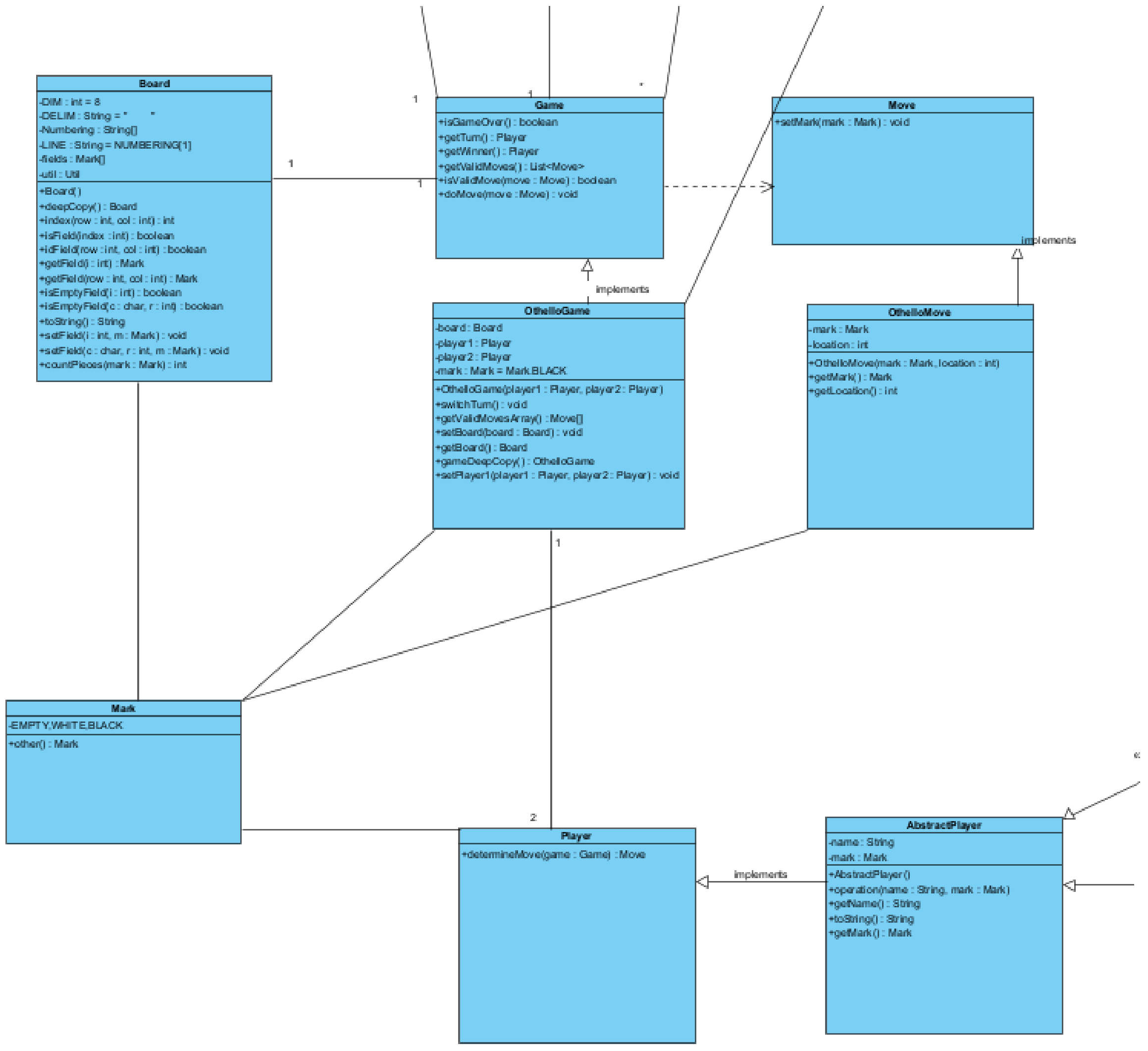


Package

- othello.model
    - game
        - Board
        - OthelloGame
        - Game
        - OthelloMove
        - Move
        - AbstractPlayer
        - Player
        - Mark
        - Util
    - server
        - OthelloServerApplication

        - ImplOthelloServer
        - OthelloServer
        - OthelloClientHandler
        - Protocol
    - client
        - Othello
        - OthelloClient
        - Client
        - OthelloListener
        - Listener
    - ai
        - ComputerPlaye
        - HardStrategy
        - EasyStrategy
        - Strategy
    - ui
        - NetworkPlayer
        - Humanplayer
        - OthelloTUI

Game


Board
-DIM : int = 8
-DELIM : String = "   "
-Numbering : String[]
-LINE : String = NUMBERING[1]
-fields : Mark[]
-util : Util
+Board()
+deepCopy() : Board
+index(row : int, col : int) : int
+isField(index : int) : boolean
+idField(row : int, col : int) : boolean
+getField(i : int) : Mark
+getField(row : int, col : int) : Mark
+isEmptyField(i : int) : boolean
+isEmptyField(c : char, r : int) : boolean
+toString() : String
+setField(i : int, m : Mark) : void
+setField(c : char, r : int, m : Mark) : void
+countPieces(mark : Mark) : int
1
1
1
1
*
Game
+isGameOver() : boolean
+getTurn() : Player
+getWinner() : Player
+getValidMoves() : List<Move>
+isValidMove(move : Move) : boolean
+doMove(move : Move) : void
Move
+setMark(mark : Mark) : void
implements
implements
OthelloGame
-board : Board
-player1 : Player
-player2 : Player
-mark : Mark = Mark.BLACK
+OthelloGame(player1 : Player, player2 : Player)
+switchTurn() : void
+getValidMovesArray() : Move[]
+setBoard(board : Board) : void
+getBoard() : Board
+gameDeepCopy() : OthelloGame
+setPlayer1(player1 : Player, player2 : Player) : void
OthelloMove
-mark : Mark
-location : int
+OthelloMove(mark : Mark, location : int)
+getMark() : Mark
+getLocation() : int
1
Mark
-EMPTY,WHITE,BLACK
+other() : Mark
2
Player
+determineMove(game : Game) : Move
implements
AbstractPlayer
-name : String
-mark : Mark
+AbstractPlayer()
+operation(name : String, mark : Mark)
+getName() : String
+toString() : String
+getMark() : Mark

Classes

- game
  - Board: checking the status of the board. initialized when the game has started. defining what the board is. It is related to the OthelloGame. When the OthelloGame is constructed, the Board also is made and used by the OthelloGame. OthelloGame has one board.
  - OthelloGame: Make exact changes to the board. Define game logic like win. Using move setting marks on the board. It is related to the Board, and Move, Player. When it is initialized, get parameters to set the player and make a new board. And move is used as a parameter for making changes. OthelloGame has one board and two players and uses Move multiple times. Games will be created at Client and Server and the server can manage multiple games.
  - Game: as an interface, it defines general methods which are necessary for the board game.
  - OthelloMove: storing Mark and index of location. This is related to the OthelloGame which uses OthelloMove to make the real move on the board.
  - Move: as an interface, it defines the usual method for a board game.

    - AbstractPlayer: This is the abstraction of Player. Defining common using methods which are determining move, getter for mark and name. This is related to OthelloGame which uses two Players for starting the game. It is an implementation of the Player interface.
    - Player: as interface, define general commands for a board game.
    - Mark: Defining a variety of marks as EMPTY, BLACK, WHITE. Has other method to return the mark. This is used in whole game classes.
    - Util: provides utility methods especially calculating from command to index. This is widely used among game classes.

Responsibilities

- Define the board
- Monitor the board
- Make changes to the board
- Provide game logic
- Store move
- Determine move
- Define mark
- Change keyword to index number
    - These responsibilities are related to FR1: A standard game can be played on both client and server in conjunction with the reference server and client, respectively.
    - These game classes make it possible to play standard games. Game logic, board itself, move, and player are all essential parts of the board game.

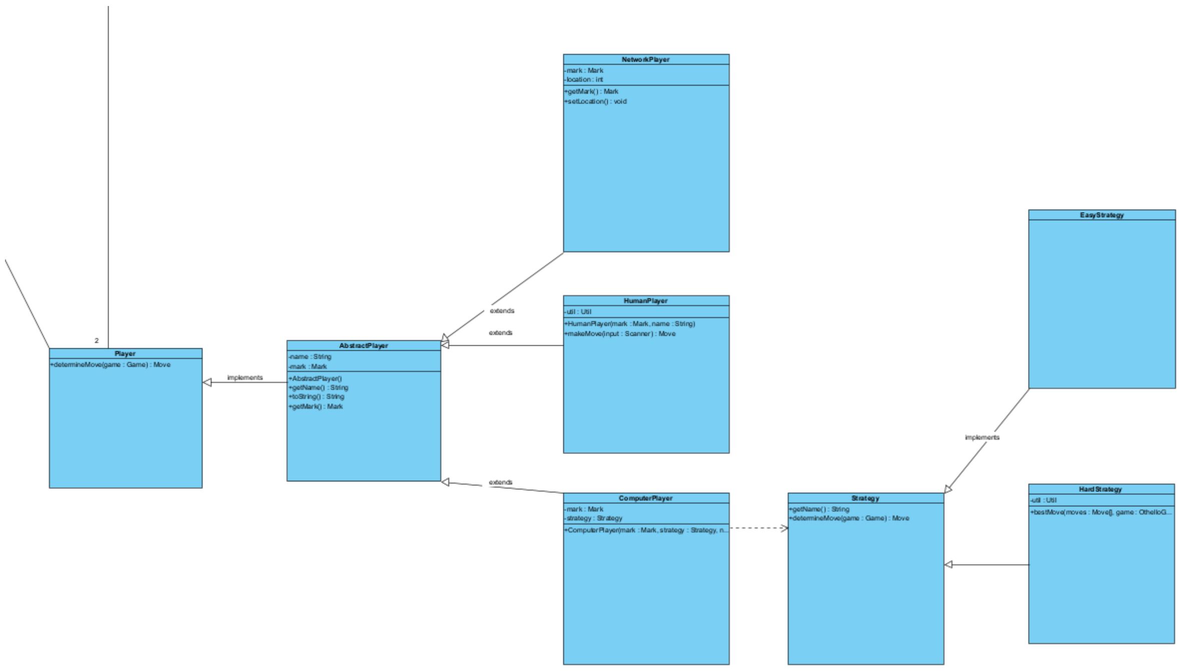
Package

- This package named “game” contains all classes for playing Othello games. Mostly, these classes are made for making the game logic of Othello work.

Design

- To achieve the MVC model of design, classes related to game logic are separated from UI, Network and AI. Then, if we have changes in network and ui, we can easily maintain it.
- OthelloGame has main functionality especially game logic, then Board is changed by methods of the OthelloGame. OthelloMove can contain specific moves for the Board and OthelloGame will use it.

Players



Classes

- game
    - AbstractPlayer: This is the abstraction of Player. Defining common using methods which are determining move, getter for mark and name. This is related to OthelloGame which uses two Players for starting the game. It is an implementation of the Player interface.
    - Player: as interface, define general commands for a board game.

- ui
    - NetworkPlayer: As extension of Abstract Player, it determines move by Protocol messages created by user input. The player can work with a client in an environment connected with the network. It is possible to receive a Protocol message to determine move.
    - HumanPlayer: As extension of Abstract Player, it determines move according to user input from user interface. Players work from user input in an environment without a network. Workable only by input from the user interface.

- ai
    - ComputerPlayer: As an extension of AbstractPlayer, it determines move by calculation depending on strategy. It is possible to work in an environment connected with a network. Using two types of strategy, it can perform at different levels of computer.
    - Strategy: As an interface of strategy, it provides common command for strategy.
    - EasyStrategy: Implementation of Strategy, determines move automatically. This strategy is a low level strategy, so it provides a random valid move to a computer player.

    - HardStrategy: As implementation of Strategy, determines move automatically. This strategy is a high level strategy. It provides the best move for the game at that time to a computer player.

Responsibilities

- Provide a player who works from user input.
- Provide a player works as AI
- AI has at least two difficulty level
    - These responsibilities are related to FR3,4,7. Player related implementation provides us choices of playing which UI play or AI play, as well as easy mode of AI and hard mode of AI.
    - We can specify the type of players which are used by OthelloGame.

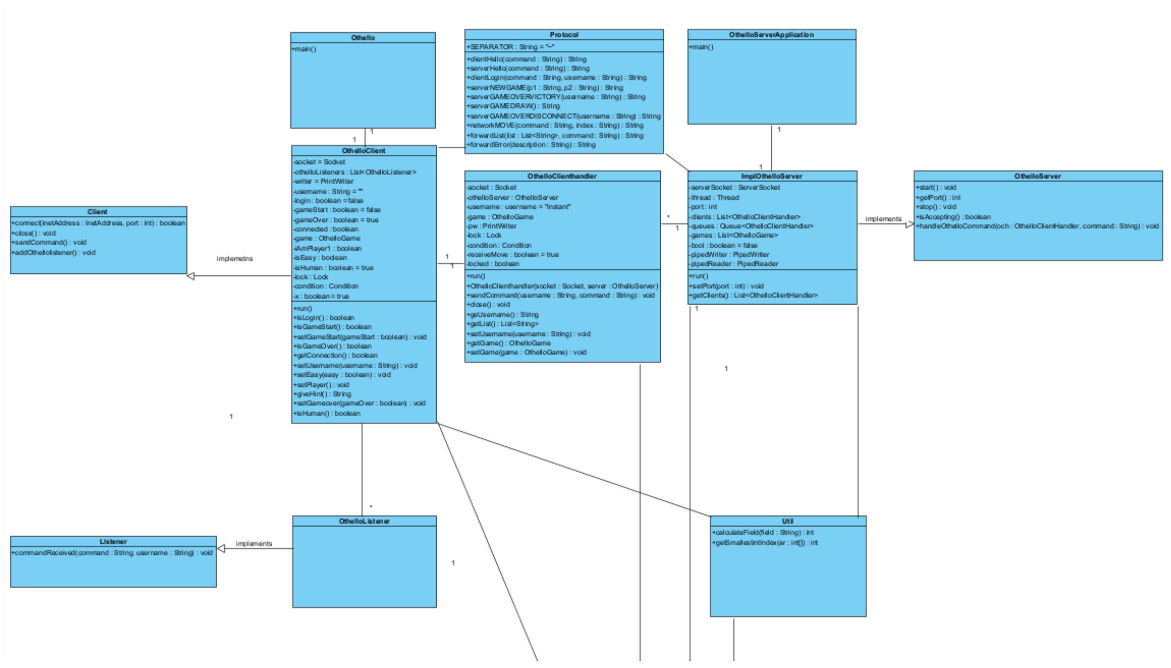
Packages

- All classes in the package of "ai" define playing as a computer. These components are not related to game logic and network, so we can easily change and make more strategies.
- Also, the "ui" package contains all classes that work for UI playing. If we need to change the ui functionality, it doesn't affect other functionality.

Design

- We chose the design of strategy. It helps us to make multiple strategies.
- Interface of Player and abstraction of Abstractplayer make our implementation less even if we need to make multiple types of player for making it fit for network playing.

Client and Server


Othello
+main()
Protocol
+SEPARATOR : String = "~"
+clientHello(command : String) : String
+serverHello(command : String) : String
+clientLogin(command : String, username : String) : String
+serverNEWGAME(p1 : String, p2 : String) : String
+serverGAMEOVERVICTORY(username : String) : String
+serverGAMEDRAW() : String
+serverGAMEOVERDISCONNECT(username : String) : String
+networkMOVE(command : String, index : String) : String
+forwardList(list : List<String>, command : String) : String
+forwardError(description : String) : String
OthelloServerApplication
+main()
OthelloClient
-socket = Socket
-othelloListeners : List<OthelloListener>
-writer = PrintWriter
-username : String = ""
-login : boolean = false
-gameStart : boolean = false
-gameOver : boolean = true
-connected : boolean
-game : OthelloGame
-iAmPlayer1 : boolean
-isEasy : boolean
-isHuman : boolean = true
-lock : Lock
-condition : Condition
-x : boolean = true
+run()
+isLogin() : boolean
+isGameStart() : boolean
+setGameStart(gameStart : boolean) : void
+isGameOver() : boolean
+getConnection() : boolean
+setUsername(username : String) : void
+setEasy(easy : boolean) : void
+setPlayer() : void
+giveHint() : String
+setGameover(gameOver : boolean) : void
+isHuman() : boolean
OthelloClienthandler
-socket : Socket
-othelloServer : OthelloServer
-username : username = "instant"
-game : OthelloGame
-pw : PrintWriter
-lock : Lock
-condition : Condition
-receiveMove : boolean = true
-locked : boolean
+run()
+OthelloClienthandler(socket : Socket, server : OthelloServer)
+sendCommand(username : String, command : String) : void
+close() : void
+getUsername() : String
+getList() : List<String>
+setUsername(username : String) : void
+getGame() : OthelloGame
+setGame(game : OthelloGame) : void
ImplOthelloServer
-serverSocket : ServerSocket
-thread : Thread
-port : int
-clients : List<OthelloClientHandler>
-queues : Queue<OthelloClientHandler>
-games : List<OthelloGame>
-bool : boolean = false
-pipedWriter : PipedWriter
-pipedReader : PipedReader
+run()
+setPort(port : int) : void
+getClients() : List<OthelloClientHandler>
implements
OthelloServer
+start() : void
+getPort() : int
+stop() : void
+isAccepting() : boolean
+handleOthelloCommand(och : OthelloClientHandler, command : String) : void
Client
+connect(inetAddress : InetAddress, port : int) : boolean
+close() : void
+sendCommand() : void
+addOthellolistener() : void
implemetns
Listener
+commandReceived(command : String, username : String) : void
implements
OthelloListener
Util
+calculateField(field : String) : int
+getSmallestIntIndex(ar : int[]) : int
1
1
1
1
1
1
*
1
1
1
1
*
1
1

Classes

- client

    - Othello: Works as user interface of OthelloClient. It allows user input for making connections with the server and all input during the game and outside the game. Ask port number and IP address to connect to a socket. Also send Queue to the server for a new game.
    - OthelloClient: This is the Client function of Othello. It starts running until the connection with the server is broken. It sends Protocol messages to the server to manage the game and process outside the game at the client and the server respectively. This class makes a connection using user input of IP address and a port number. The client has the game after receiving a new game message from the server to work independently.
    - Client: As an interface of Client, it provides common commands for working as a client in networking development.
    - OthelloListener: According to the listener model, this is working as a listener which sends and shows messages from server instead of clients. It provides us flexibility in terms of implementation.
    - Listener: As interface of Listener, provides common method as listener. That is sending messages to all clients which they have listeners.

- server
    - OthelloServerApplication: This is User interface for OthelloServer. It is needed to get input as a port number from the user. Ask again until it gets the port number which has not been used yet.
    - ImplOthelloServer: It provides the server function for a Othello game with network connection. Make a server socket by a port number as input from the user. Will be connected by several clients and communicate with them.
    - OthelloServer: As the interface of a server, it provides general operations for working as a server with network connection. Has multiple Clients as connection and makes client handlers for each client.
    - OthelloClientHandler: It is created by the server to communicate with clients respectively. Receive messages from client and send to the server and also send commands from the server to clients.
    - Protocol: Provides a function which changes command to Protocol type of messages which are sufficiently used for communication between Clients and the server at the network.

Responsibilities

- OthelloClient can start and play a standard game.
- ImplOthelloServer can start and play a standard game.
- A server starts using a port number by user input.
- A server asks again if the port is used already.
- A client starts using IP address and port number by user input. And connect to a socket.
- Able to start a new game immediately after the game is ended.
- All communication outside of the game works on client and server in conjunction with the reference server and client, respectively.
- Whenever the client loses connection to the server, the client will be closed.
- Client loses connection during the game, the server informs the other client and this client can start a new game immediately.

    - Although implementation of Client and Server are related to all Functional requirements, especially these responsibilities satisfy FR1,4,5,9,10,11,12.
    - These provide running clients and the server, conjunction of each client and the server, User interface and pop up new games in convenience.

Packages

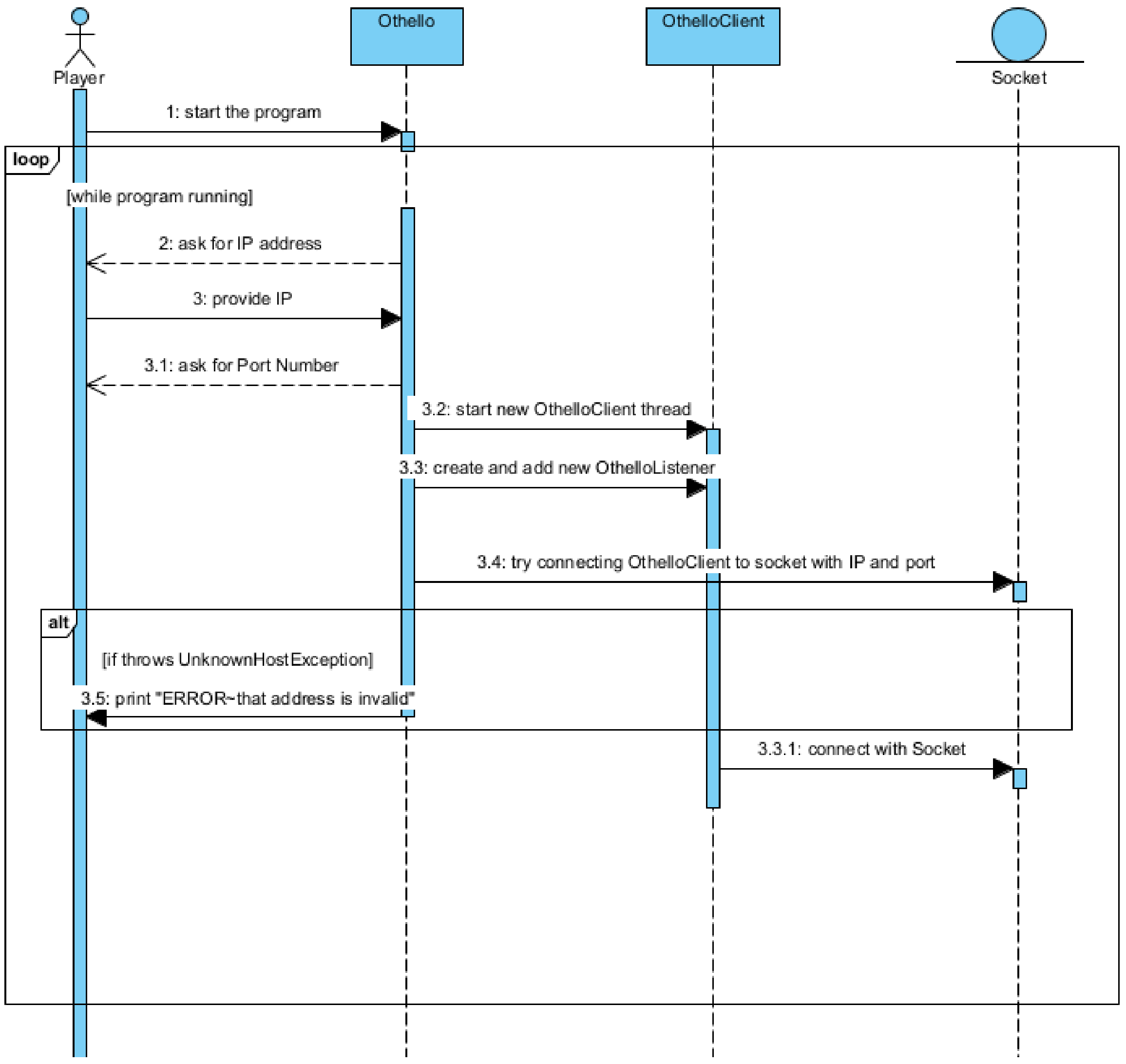
- All client related classes are in a “client” package. Provides User Interface for getting input for users and also gives them client functionality. The client works as a command and the listener works as a view in the client implementation.
- “server” package provides all server related functions including Protocol and a client handler. It can construct the server and works as a server with clients through the client handler.
- This package separation allows us to develop separately in the server function and the client function.

Design

- Listener pattern is applied to the client implementation. Exact works and showing results are separated.
- We applied connections among the server , the client handler and the client. Functionality for achieving the network communication is separated in each class.

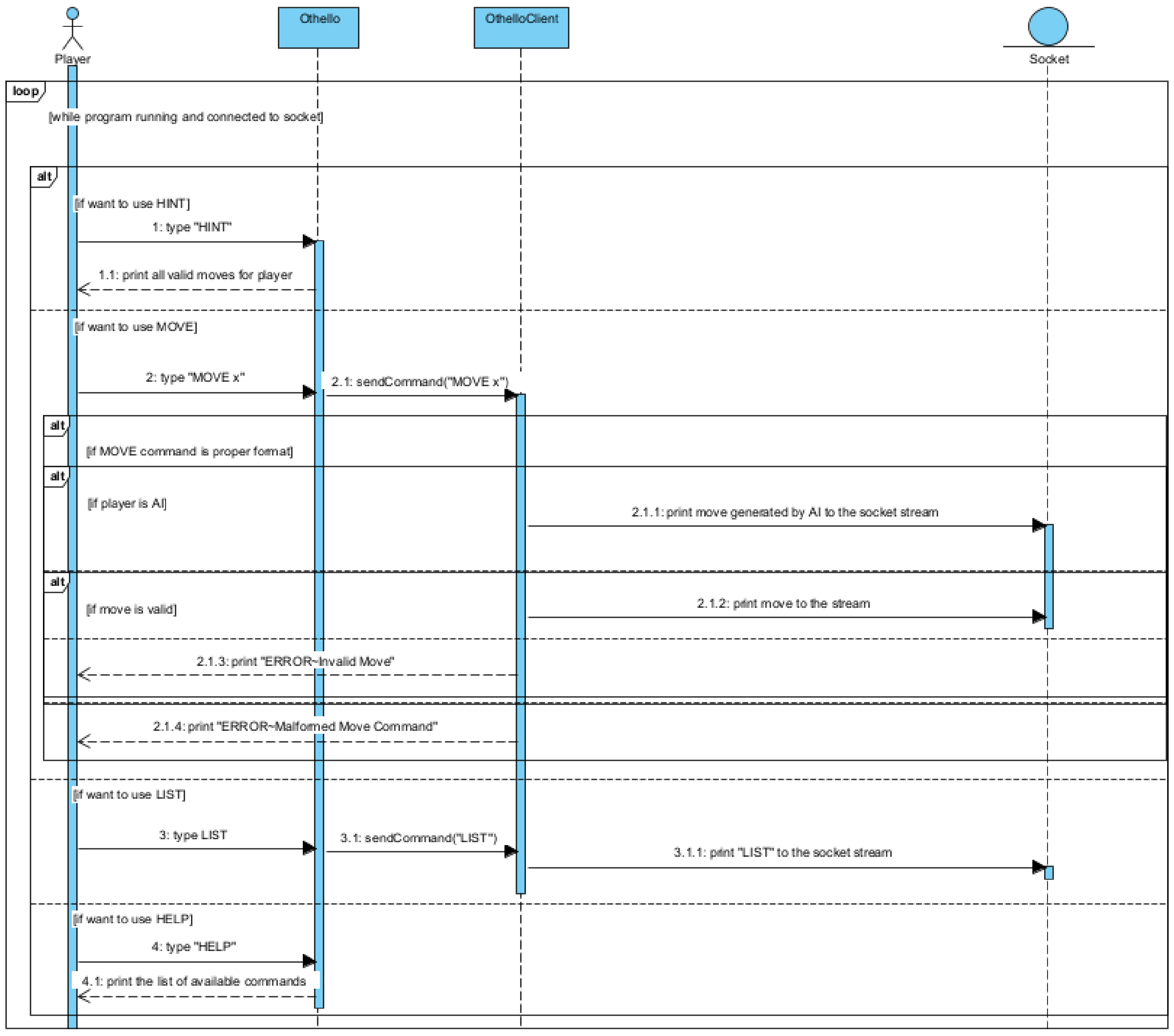
Sequence Diagram

Execution flow


Player
Othello
OthelloClient
Socket
1: start the program
loop
[while program running]
2: ask for IP address
3: provide IP
3.1: ask for Port Number
3.2: start new OthelloClient thread
3.3: create and add new OthelloListener
3.4: try connecting OthelloClient to socket with IP and port
alt
[if throws UnknownHostException]
3.5: print "ERROR~that address is invalid"
3.3.1: connect with Socket

Description:
The diagram displays the process our system uses to connect a client to the server. Firstly the user starts the UI for the client. Once it begins running the UI asks for an IP address and the port number they would like to connect to. Once that is done, a new OthelloClient thread is started and the UI provides information to the client which uses it to connect to the server. If the provided information is correct a connection is established, else a UnknownHostException is thrown.

Execution flow


Player
Othello
OthelloClient
Socket
loop
[while program running and connected to socket]
alt
[if want to use HINT]
1: type "HINT"
1.1: print all valid moves for player
[if want to use MOVE]
2: type "MOVE x"
2.1: sendCommand("MOVE x")
alt
[if MOVE command is proper format]
alt
[if player is AI]
2.1.1: print move generated by AI to the socket stream
alt
[if move is valid]
2.1.2: print move to the stream
2.1.3: print "ERROR~Invalid Move"
2.1.4: print "ERROR~Malformed Move Command"
[if want to use LIST]
3: type LIST
3.1: sendCommand("LIST")
3.1.1: print "LIST" to the socket stream
[if want to use HELP]
4: type "HELP"
4.1: print the list of available commands

Description:
The diagram describes the loop of a player sending commands to the server. The hint command only returns a list of moves from the local game. A move command has many different branches. If the format of the message is correct then the move is sent to the socket, either one generated by a computer or the one made by the player. If these conditions are not met the user gets an error of either Invalid Move or Malformed Move Command. “LIST” simply sends “LIST” to the socket and “HELP” prints a list of all the available commands locally.

Concurrency mechanism

When the thread is created

- Server makes the thread itself when they are started.

Server makes its own thread, because each server must work even if there are multiple servers. Each server will hold their own socket, client handlers and clients.

- Server makes the thread of Client Handler when Client connects to the Socket.

Whenever the server accepts connection to the socket, the server makes a client handler. The new thread of client handlers works for their own Client who made a connection to the server socket.
For giving functional support for each client's thread, the client handler also has threads for each client.

- Client makes the thread itself when they successfully connect to the Socket.

When the client connects to the server socket, they make the thread for themselves. It allows multiple clients to connect to the server and works individually by input from its own user.

Which fields must be accessed by the thread

- Server

clients: List<OthelloClientHandler> : When the server is connected with clients and it creates a client handler, it is accessed by thread of the server. It is shared among threads of the server to check who logged in.
queues: Queue<OthelloClientHandler> : When clients are logged in, their client handlers are going to the queue. It is shared among the server for checking who is going to play for the OthelloGame.
games: List<OthelloGame>: When more than two clients are in a queue, the server created the Othello game. It is shared among servers for the operation.

- Client Handler

receiveMove: boolean: it is shared among client handlers to check their messages are going to the server and it sends them back to the client handler. It stops receiving commands from the client until it is handled.

- Client

x: boolean: it is shared among threads of clients. It is used to check all commands are sended and handled at the server.

The mechanisms

User input is consumed by the client. They send commands to the client handler using writer. Client handlers get these messages through the reader and send these commands to the server, then the server will work by this input and send back to the client handler. Finally,

the client handler writes these messages to send to the client. The client reads these messages using a reader.
Concurrency issues happen when multiple threads want to change some variables in specific classes like the server concurrently.

For example, it can happen if two threads want to put the client handler to the queue list for playing games at the same time. The result of the queue list is possibly less than ideal, as only one of them is successful to be in the queue.

We made the lock and the condition at the client and client handler. In terms of the client, it works by keeping the thread waiting until all commands are sent to the server and the client receives all sending back commands from the server.
And also, the lock and the condition at the client handler helps functionality to keep threads waiting by commands being sent to the server and changed at the server for forwarding these messages to the client.

What is potential issues

It can go wrong when clients send commands multiple times at the same time to the server. We tried to make the server thread safe, but locks in Client Handler are not perfectly working, so sometimes Clients can send commands at the same time.
Then, it makes the server work hard if commands come before the server changes the board of the game. Ideally, we should make the client handler wait to send until the server finishes working.

Reflection on design

Initial Design Red 14

Initial Design

Pros

- Making lists of classes really help us to check what we need in total
- Functional and non - functional requirements are helpful during development.
- Class design helps us to recognize how classes are related.
- Sequence diagrams describe how systems work step by step and it helps to make the test case.

Cons

- The function of each class should have been described.
- Sequence diagrams should separate depending on parts of the whole functionality.
- Concurrency issues should be predicted more clearly.
- All planning of both test and coding should have been combined. We had better make a combined schedule for finishing the project.

- It is more clear to make lists of responsibility following the functional requirements. It can help us prioritize tasks.

We can develop a much better initial design at the next project for making a Test plan specifically. We made a little simple plan. The specific test case is really helpful for making ideal functionality and checking code one by one.

The design process of the initial design was poor. Firstly, we have to prioritize parts of design. Then, making scope or target of each design should help time management and making good quality design.

Final Design

Reflection on Final Design:

Firstly, the design as a whole could be way more modular. When we started with our initial design we had no idea of the scope and complexity of the project as neither of us are experienced programmers and the lab exercises did not prepare us for this many levels of abstraction. Because of that we often found ourselves confused by extremely long and convoluted classes and functions that took a lot of time to decipher for the other person that didn’t create them.

Secondly, we should have started thinking about the concurrency way more before even starting any code as in the end solving all the bugs that involved thread safety stole around 3 days of our time. We could probably have looked into pre-established models like the producer/consumer one and adapted the system around that.

Another design flaw we believe could be implemented better is the interaction of AI players with the client/server. Since we chose for the system to recognize the computer player “MOVE” queries with a 100 which is just a chosen integer, it could lead to errors if perhaps a human player tries it and our checks fail in sendCommand. Perhaps the process of “playing” should be separated and have different implementations for humans and AI.

Lastly we should have completely gotten rid of any functionality from User Interfaces for both the server and the client, since having it there made the line between the interface and the class it operated on very blurred and created unnecessary strain on both implementing new functionality as well as the debugging of any arising errors.

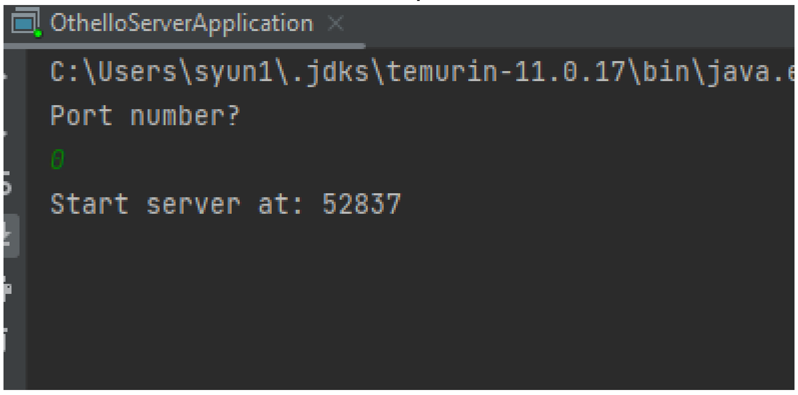
System tests

Testing Report for FR1
FR1: Standard game
Expected behavior: - A standard game can be played on both client and server in conjunction with the reference server and client, respectively. - MOVE~N command can be sent correctly from the server to the client and the client to the server.

- Both game boards at the client and the server can be changed by messages respectively.

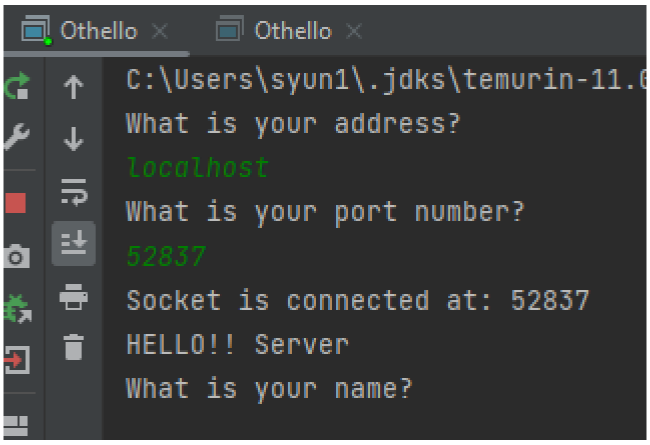
Testing result

1. the server access to the port.



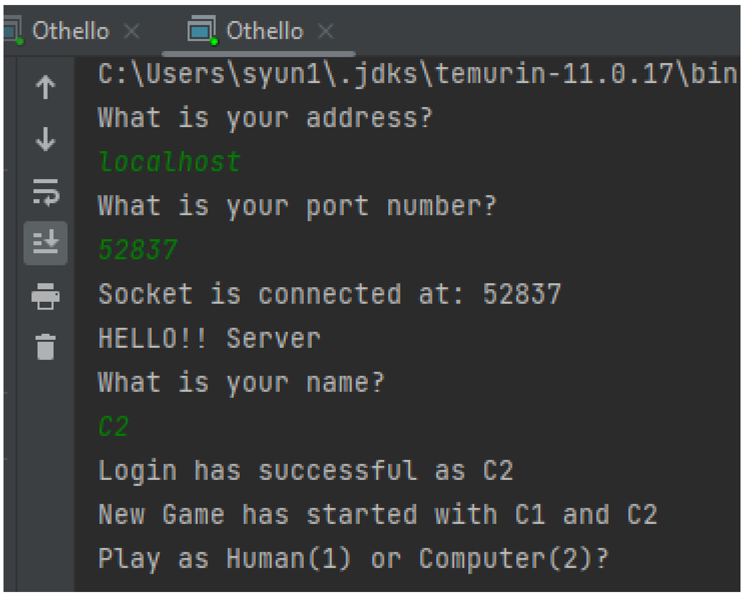
```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.e
Port number?
0
Start server at: 52837
```

2. 2 clients access the socket.

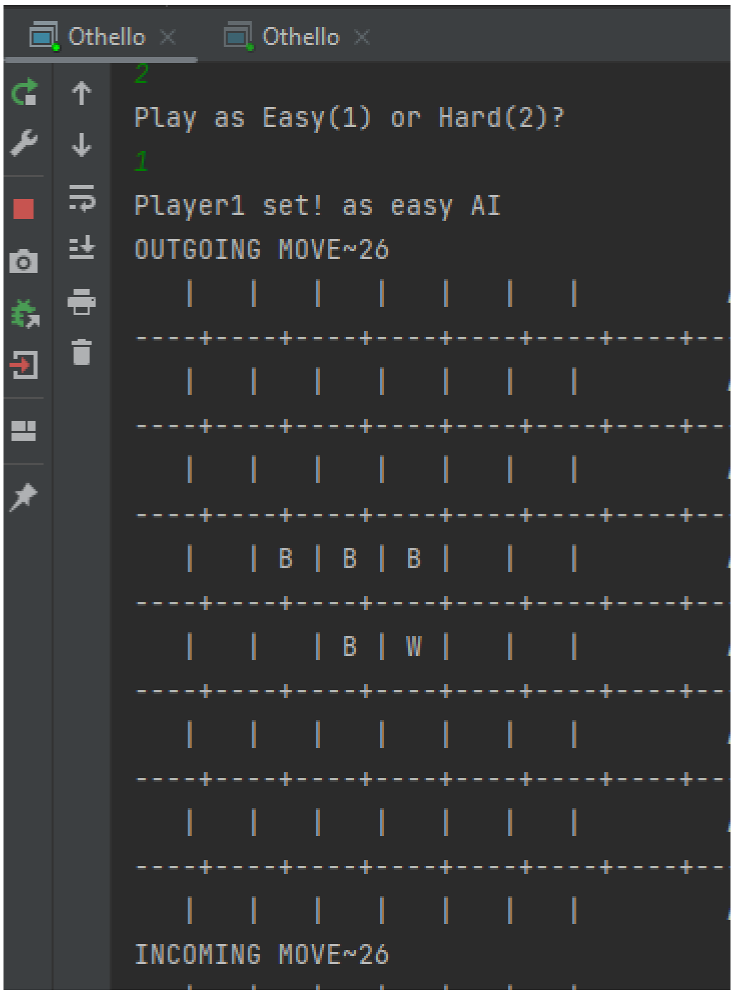


```
C:\Users\syun1\.jdks\temurin-11.0
What is your address?
localhost
What is your port number?
52837
Socket is connected at: 52837
HELLO!! Server
What is your name?
```

3. Put MOVE from one of the clients.

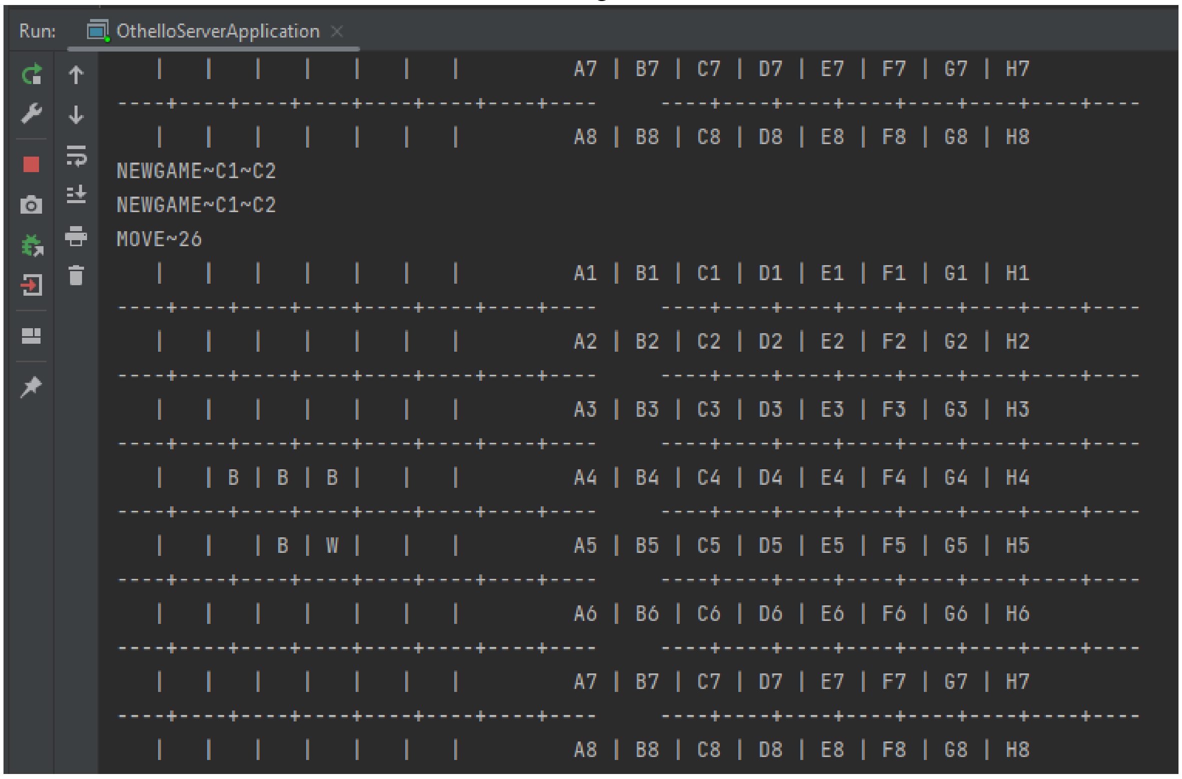


```
C:\Users\syun1\.jdks\temurin-11.0.17\bin
What is your address?
localhost
What is your port number?
52837
Socket is connected at: 52837
HELLO!! Server
What is your name?
C2
Login has successful as C2
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```



```
2
Play as Easy(1) or Hard(2)?
1
Player1 set! as easy AI
OUTGOING MOVE~26
    |    |    |    |    |    |    |
----+----+----+----+----+----+----+--
    |    |    |    |    |    |    |
----+----+----+----+----+----+----+--
    |    |    |    |    |    |    |
----+----+----+----+----+----+----+--
    |    | B  | B  | B  |    |    |
----+----+----+----+----+----+----+--
    |    |    | B  | W  |    |    |
----+----+----+----+----+----+----+--
    |    |    |    |    |    |    |
----+----+----+----+----+----+----+--
    |    |    |    |    |    |    |
----+----+----+----+----+----+----+--
    |    |    |    |    |    |    |
INCOMING MOVE~26
```

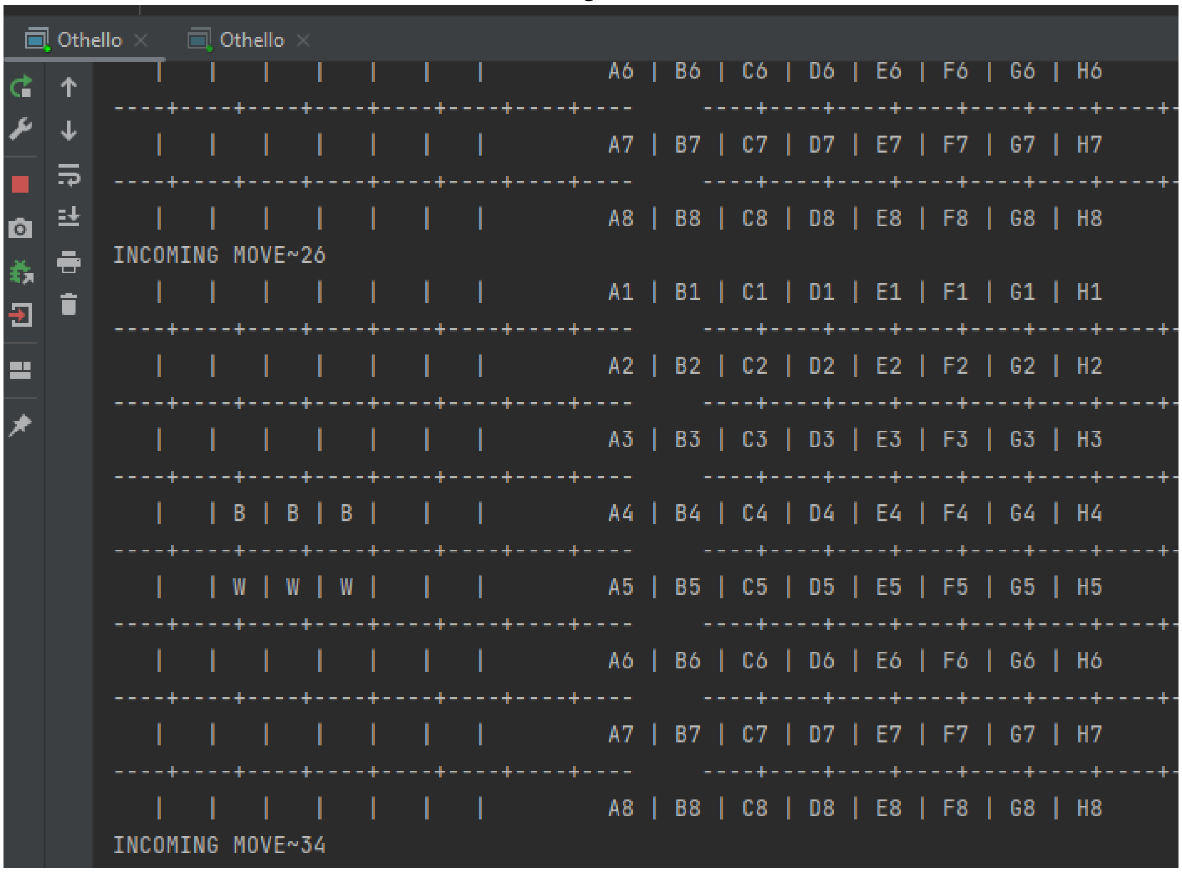
4. The server receives the correct message.



Run: OthelloServerApplication

```
   |   |   |   |   |   |   |        A7 | B7 | C7 | D7 | E7 | F7 | G7 | H7
---+---+---+---+---+---+---+---    ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |        A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
NEWGAME~C1~C2
NEWGAME~C1~C2
MOVE~26
   |   |   |   |   |   |   |        A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
---+---+---+---+---+---+---+---    ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |        A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
---+---+---+---+---+---+---+---    ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |        A3 | B3 | C3 | D3 | E3 | F3 | G3 | H3
---+---+---+---+---+---+---+---    ----+----+----+----+----+----+----+----
   |   | B | B | B |   |   |        A4 | B4 | C4 | D4 | E4 | F4 | G4 | H4
---+---+---+---+---+---+---+---    ----+----+----+----+----+----+----+----
   |   |   | B | W |   |   |        A5 | B5 | C5 | D5 | E5 | F5 | G5 | H5
---+---+---+---+---+---+---+---    ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |        A6 | B6 | C6 | D6 | E6 | F6 | G6 | H6
---+---+---+---+---+---+---+---    ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |        A7 | B7 | C7 | D7 | E7 | F7 | G7 | H7
---+---+---+---+---+---+---+---    ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |        A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
```

5. The client receives the correct message.



```
    |    |    |    |    |    |    |         A6 | B6 | C6 | D6 | E6 | F6 | G6 | H6
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+-
    |    |    |    |    |    |    |         A7 | B7 | C7 | D7 | E7 | F7 | G7 | H7
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+-
    |    |    |    |    |    |    |         A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
INCOMING MOVE~26
    |    |    |    |    |    |    |         A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+-
    |    |    |    |    |    |    |         A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+-
    |    |    |    |    |    |    |         A3 | B3 | C3 | D3 | E3 | F3 | G3 | H3
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+-
    |    | B  | B  | B  |    |    |         A4 | B4 | C4 | D4 | E4 | F4 | G4 | H4
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+-
    |    | W  | W  | W  |    |    |         A5 | B5 | C5 | D5 | E5 | F5 | G5 | H5
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+-
    |    |    |    |    |    |    |         A6 | B6 | C6 | D6 | E6 | F6 | G6 | H6
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+-
    |    |    |    |    |    |    |         A7 | B7 | C7 | D7 | E7 | F7 | G7 | H7
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+-
    |    |    |    |    |    |    |         A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
INCOMING MOVE~34
```

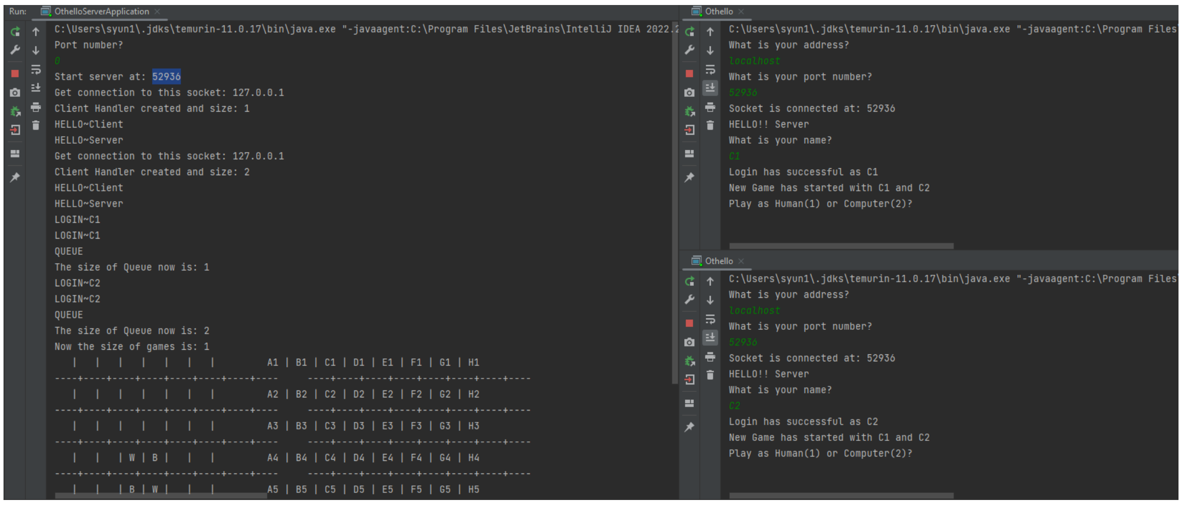
Both the server and the client send/ receive commands correctly. It is possible to play a standard game.

Testing Report for FR2

FR2: Play as human player

Expected behavior:
- The client can play (on a server) as a human player, controlled by the user.
- The client correctly plays as a human player until the game is finished.

Testing result
1. Both the server and two clients start.

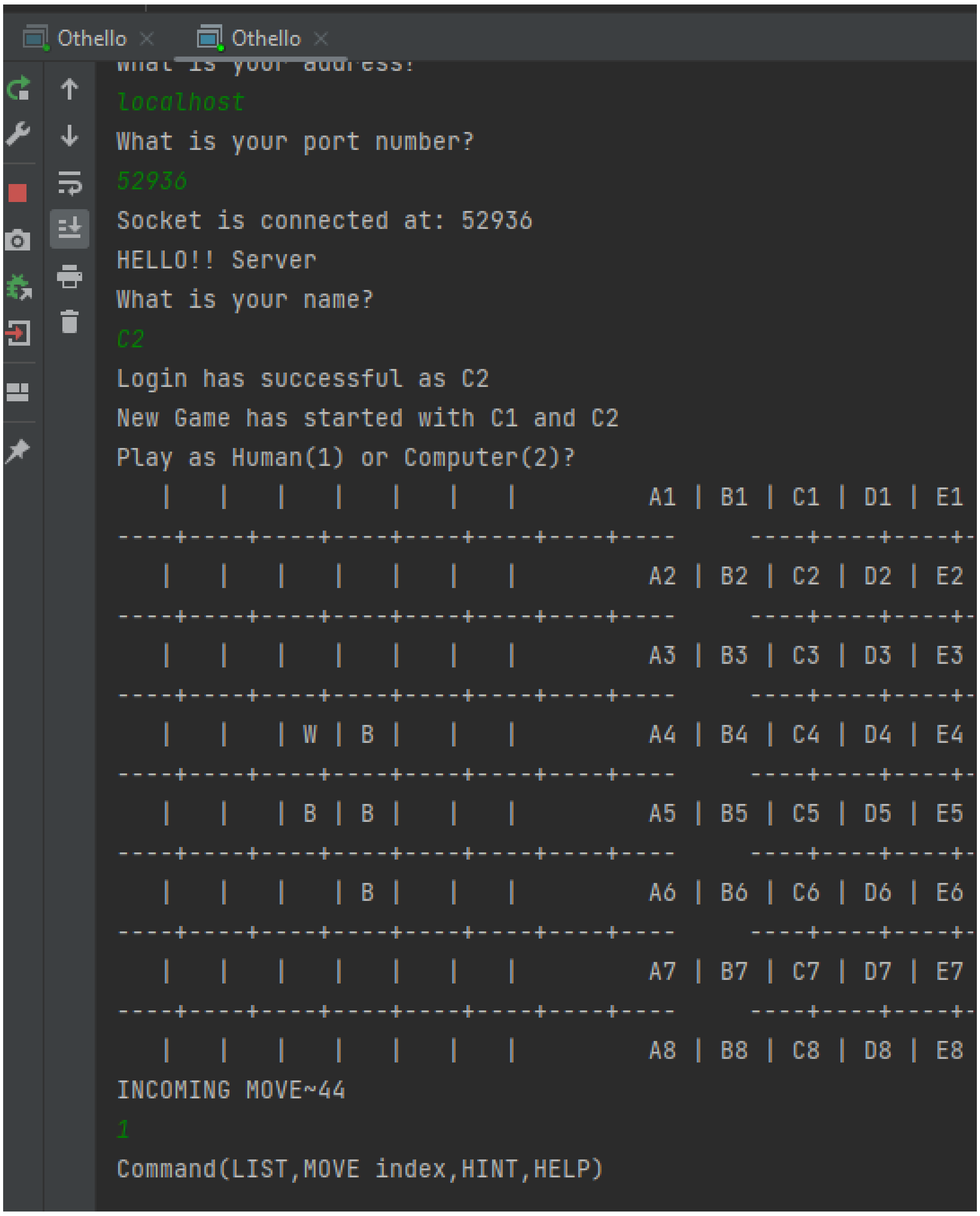


```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2
Port number?
0
Start server at: 52936
Get connection to this socket: 127.0.0.1
Client Handler created and size: 1
HELLO~Client
HELLO~Server
Get connection to this socket: 127.0.0.1
Client Handler created and size: 2
HELLO~Client
HELLO~Server
LOGIN~C1
LOGIN~C1
QUEUE
The size of Queue now is: 1
LOGIN~C2
LOGIN~C2
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
    |    |    |    |    |    |    |         A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A3 | B3 | C3 | D3 | E3 | F3 | G3 | H3
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
    |    |    | W  | B  |    |    |         A4 | B4 | C4 | D4 | E4 | F4 | G4 | H4
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
    |    |    | B  | W  |    |    |         A5 | B5 | C5 | D5 | E5 | F5 | G5 | H5
```

```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "-javaagent:C:\Program Files
What is your address?
localhost
What is your port number?
52936
Socket is connected at: 52936
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

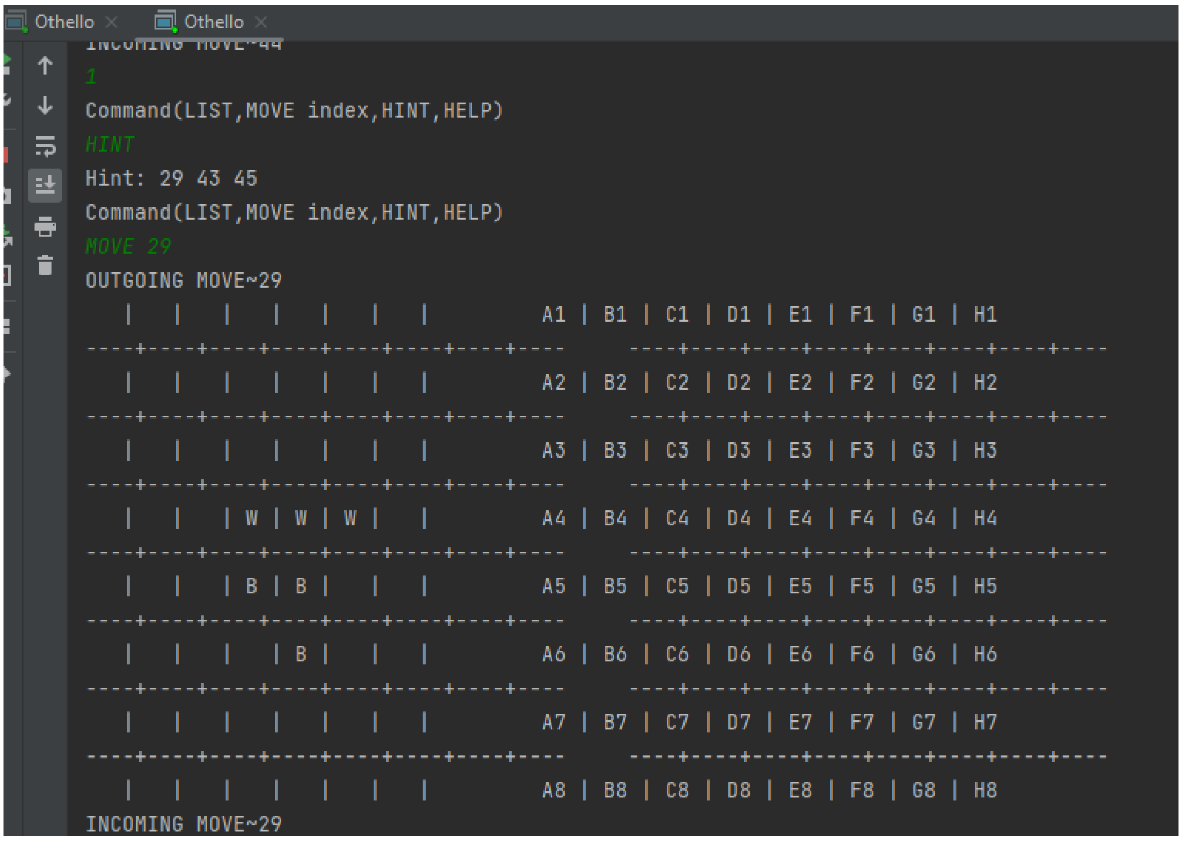
```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "-javaagent:C:\Program Files
What is your address?
localhost
What is your port number?
52936
Socket is connected at: 52936
HELLO!! Server
What is your name?
C2
Login has successful as C2
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

2. One of the clients selects a Human player to play.



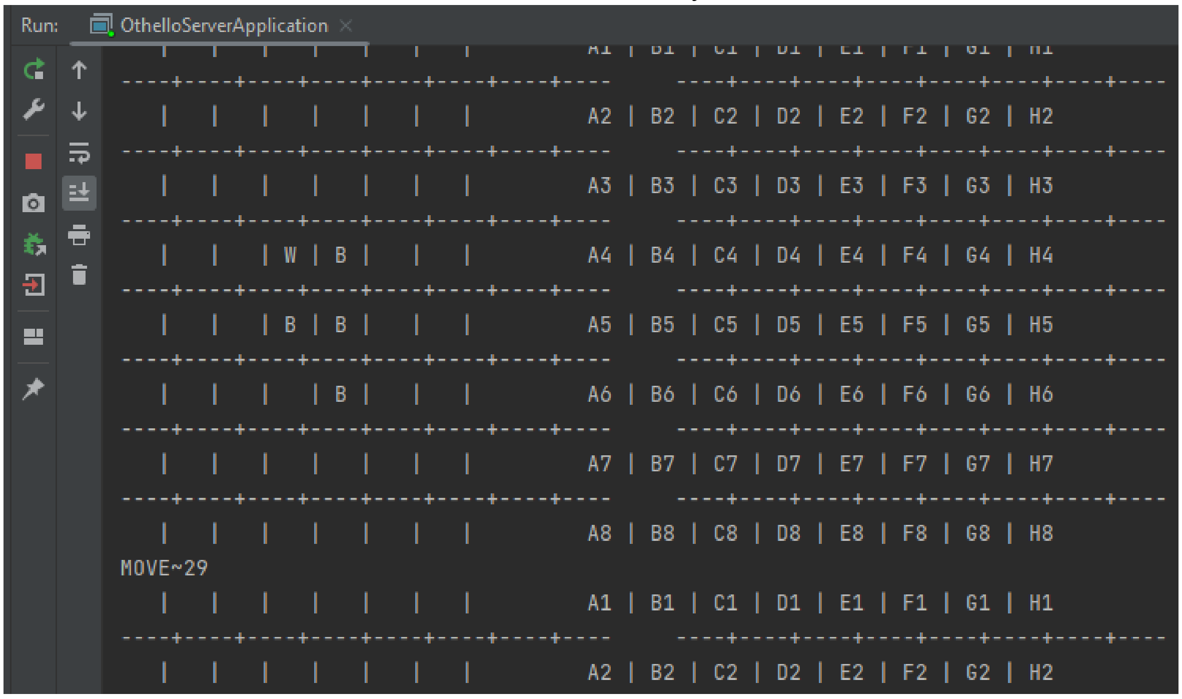
```
What is your address?
localhost
What is your port number?
52936
Socket is connected at: 52936
HELLO!! Server
What is your name?
C2
Login has successful as C2
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
   |   |   |   |   |   |   |          A1 | B1 | C1 | D1 | E1
---+---+---+---+---+---+---+---       ---+---+---+-
   |   |   |   |   |   |   |          A2 | B2 | C2 | D2 | E2
---+---+---+---+---+---+---+---       ---+---+---+-
   |   |   |   |   |   |   |          A3 | B3 | C3 | D3 | E3
---+---+---+---+---+---+---+---       ---+---+---+-
   |   |   | W | B |   |   |          A4 | B4 | C4 | D4 | E4
---+---+---+---+---+---+---+---       ---+---+---+-
   |   |   | B | B |   |   |          A5 | B5 | C5 | D5 | E5
---+---+---+---+---+---+---+---       ---+---+---+-
   |   |   |   | B |   |   |          A6 | B6 | C6 | D6 | E6
---+---+---+---+---+---+---+---       ---+---+---+-
   |   |   |   |   |   |   |          A7 | B7 | C7 | D7 | E7
---+---+---+---+---+---+---+---       ---+---+---+-
   |   |   |   |   |   |   |          A8 | B8 | C8 | D8 | E8
INCOMING MOVE~44
1
Command(LIST,MOVE index,HINT,HELP)
```

3. Conjunction between the client and the server is successful, especially sending MOVE commands.



```
INCOMING MOVE~44
1
Command(LIST,MOVE index,HINT,HELP)
HINT
Hint: 29 43 45
Command(LIST,MOVE index,HINT,HELP)
MOVE 29
OUTGOING MOVE~29
    |    |    |    |    |    |    |             A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |             A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |             A3 | B3 | C3 | D3 | E3 | F3 | G3 | H3
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    | W  | W  | W  |    |             A4 | B4 | C4 | D4 | E4 | F4 | G4 | H4
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    | B  | B  |    |    |             A5 | B5 | C5 | D5 | E5 | F5 | G5 | H5
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    | B  |    |    |             A6 | B6 | C6 | D6 | E6 | F6 | G6 | H6
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |             A7 | B7 | C7 | D7 | E7 | F7 | G7 | H7
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |             A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
INCOMING MOVE~29
```

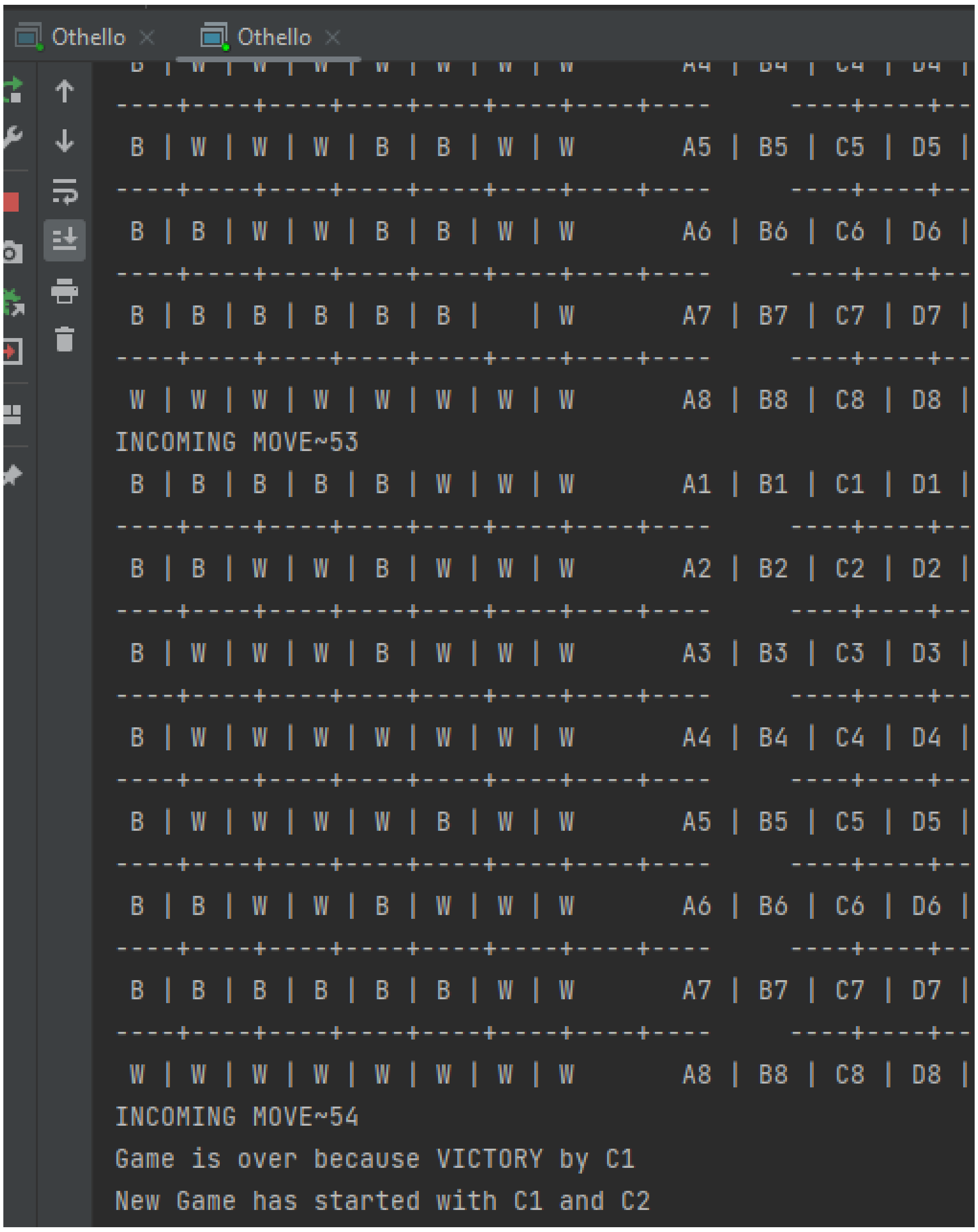
The client sends and receives commands successfully.



```
    |    |    |    |    |    |    |             A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |             A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |             A3 | B3 | C3 | D3 | E3 | F3 | G3 | H3
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    | W  | B  |    |    |             A4 | B4 | C4 | D4 | E4 | F4 | G4 | H4
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    | B  | B  |    |    |             A5 | B5 | C5 | D5 | E5 | F5 | G5 | H5
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    | B  |    |    |             A6 | B6 | C6 | D6 | E6 | F6 | G6 | H6
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |             A7 | B7 | C7 | D7 | E7 | F7 | G7 | H7
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |             A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
MOVE~29
    |    |    |    |    |    |    |             A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |             A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
```

The server also receives and sends correct messages.

4. Play until the game is finished.



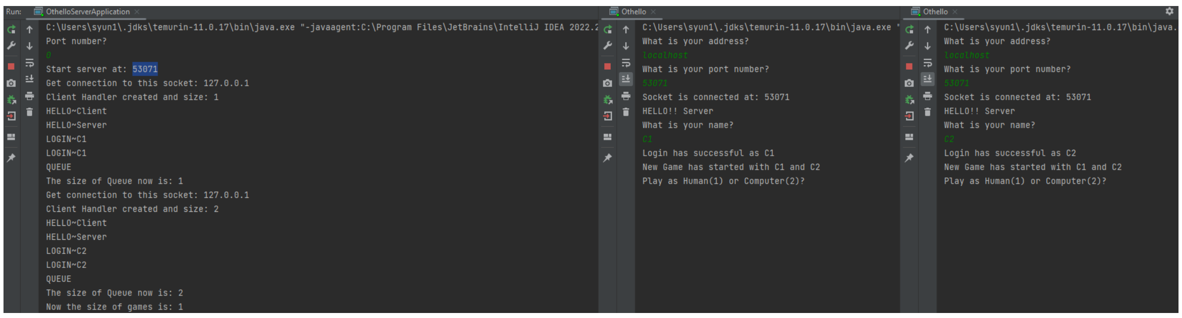
```
 B  | W  | W  | W  | W  | W  | W  | W        A4  | B4  | C4  | D4  |
----+----+----+----+----+----+----+----      ----+----+--
 B  | W  | W  | W  | B  | B  | W  | W        A5  | B5  | C5  | D5  |
----+----+----+----+----+----+----+----      ----+----+--
 B  | B  | W  | W  | B  | B  | W  | W        A6  | B6  | C6  | D6  |
----+----+----+----+----+----+----+----      ----+----+--
 B  | B  | B  | B  | B  | B  |    | W        A7  | B7  | C7  | D7  |
----+----+----+----+----+----+----+----      ----+----+--
 W  | W  | W  | W  | W  | W  | W  | W        A8  | B8  | C8  | D8  |
INCOMING MOVE~53
 B  | B  | B  | B  | B  | W  | W  | W        A1  | B1  | C1  | D1  |
----+----+----+----+----+----+----+----      ----+----+--
 B  | B  | W  | W  | B  | W  | W  | W        A2  | B2  | C2  | D2  |
----+----+----+----+----+----+----+----      ----+----+--
 B  | W  | W  | W  | B  | W  | W  | W        A3  | B3  | C3  | D3  |
----+----+----+----+----+----+----+----      ----+----+--
 B  | W  | W  | W  | W  | W  | W  | W        A4  | B4  | C4  | D4  |
----+----+----+----+----+----+----+----      ----+----+--
 B  | W  | W  | W  | W  | B  | W  | W        A5  | B5  | C5  | D5  |
----+----+----+----+----+----+----+----      ----+----+--
 B  | B  | W  | W  | B  | W  | W  | W        A6  | B6  | C6  | D6  |
----+----+----+----+----+----+----+----      ----+----+--
 B  | B  | B  | B  | B  | B  | W  | W        A7  | B7  | C7  | D7  |
----+----+----+----+----+----+----+----      ----+----+--
 W  | W  | W  | W  | W  | W  | W  | W        A8  | B8  | C8  | D8  |
INCOMING MOVE~54
Game is over because VICTORY by C1
New Game has started with C1 and C2
```

Testing Report for FR3

FR3: Play as AI

Expected behavior:
- The client can play (on a server) as a computer player, controlled by AI.
- The client can play as a computer player until the game is finished.

Testing result
1. Both the server and two clients start.

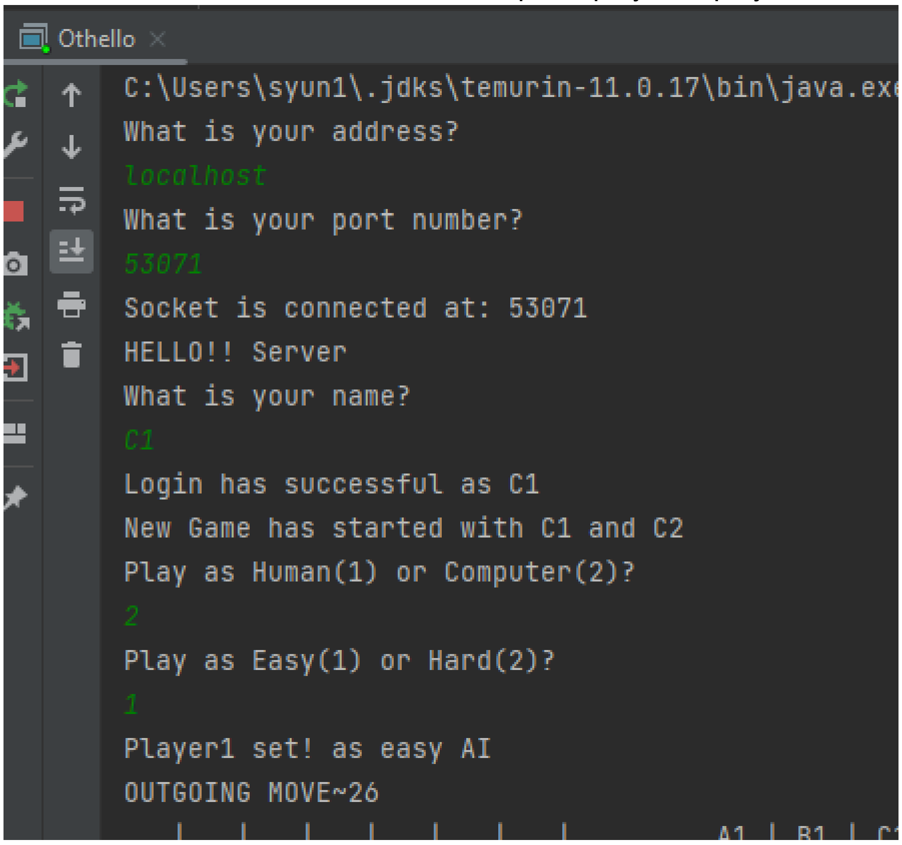


```
Run: OthelloServerApplication
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2
Port number?
0
Start server at: 53071
Get connection to this socket: 127.0.0.1
Client Handler created and size: 1
HELLO~Client
HELLO~Server
LOGIN~C1
LOGIN~C1
QUEUE
The size of Queue now is: 1
Get connection to this socket: 127.0.0.1
Client Handler created and size: 2
HELLO~Client
HELLO~Server
LOGIN~C2
LOGIN~C2
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
```

```
Othello
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "
What is your address?
localhost
What is your port number?
53071
Socket is connected at: 53071
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

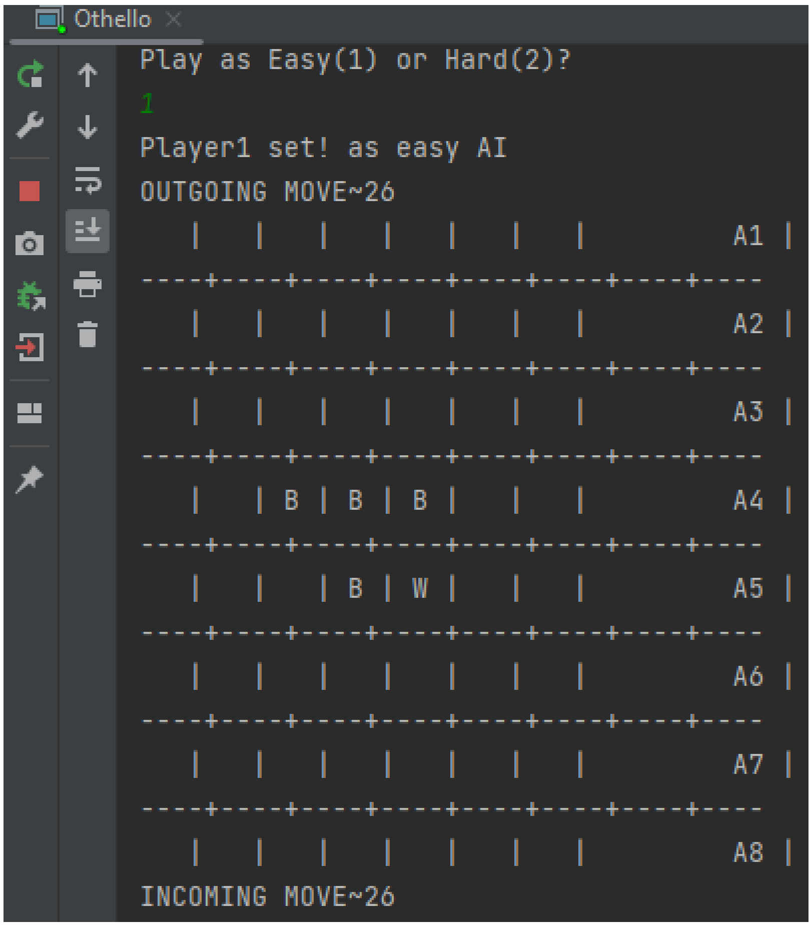
```
Othello
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.
What is your address?
localhost
What is your port number?
53071
Socket is connected at: 53071
HELLO!! Server
What is your name?
C2
Login has successful as C2
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

2. One of the clients selects a Computer player to play.

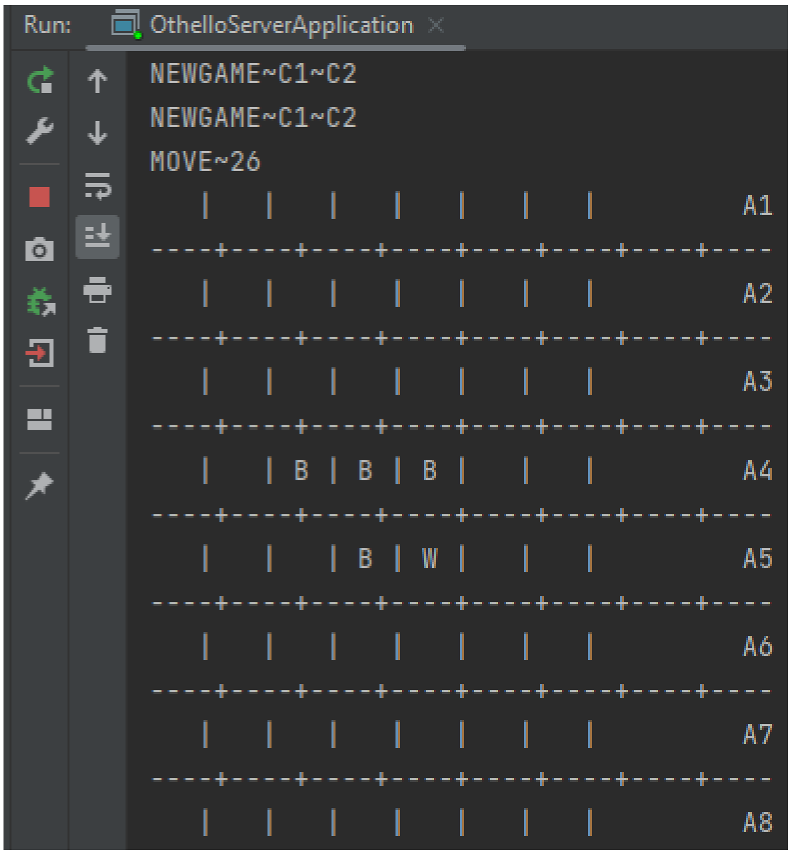


```
Othello
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
What is your address?
localhost
What is your port number?
53071
Socket is connected at: 53071
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
2
Play as Easy(1) or Hard(2)?
1
Player1 set! as easy AI
OUTGOING MOVE~26
    |   |   |   |   |   |   |       A1  |  B1  |  C
```

3. MOVE commands are correctly sent and received on both the server and the client.

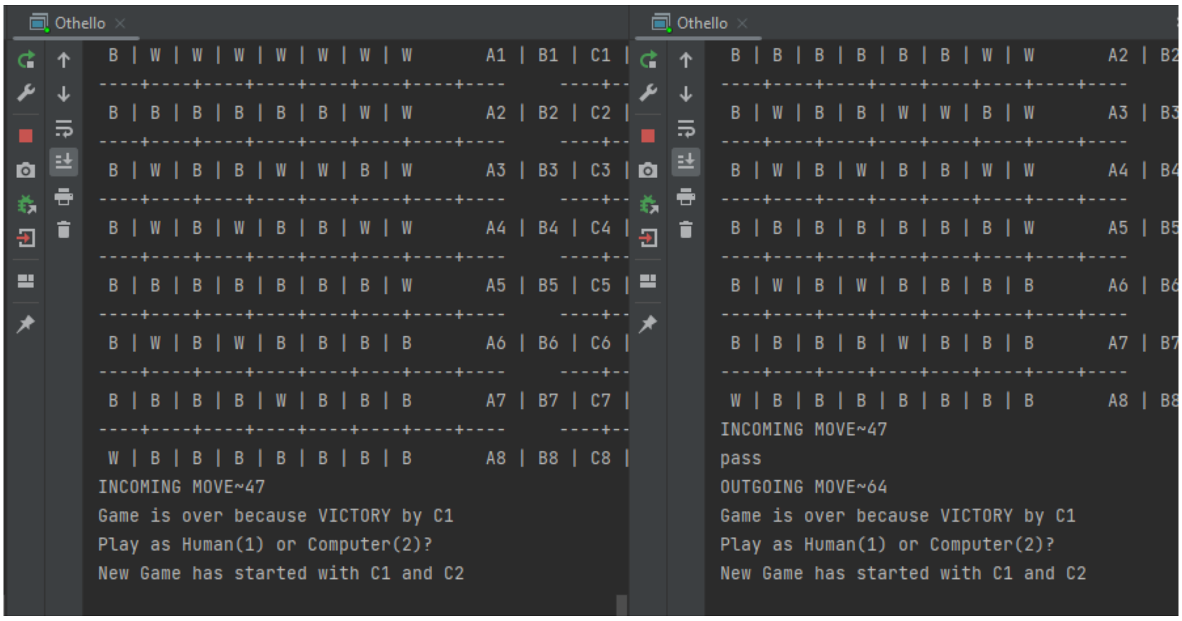


```
Play as Easy(1) or Hard(2)?
1
Player1 set! as easy AI
OUTGOING MOVE~26
    |    |    |    |    |    |    |         A1  |
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A2  |
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A3  |
----+----+----+----+----+----+----+----
    |    | B  | B  | B  |    |    |         A4  |
----+----+----+----+----+----+----+----
    |    |    | B  | W  |    |    |         A5  |
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A6  |
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A7  |
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A8  |
INCOMING MOVE~26
```



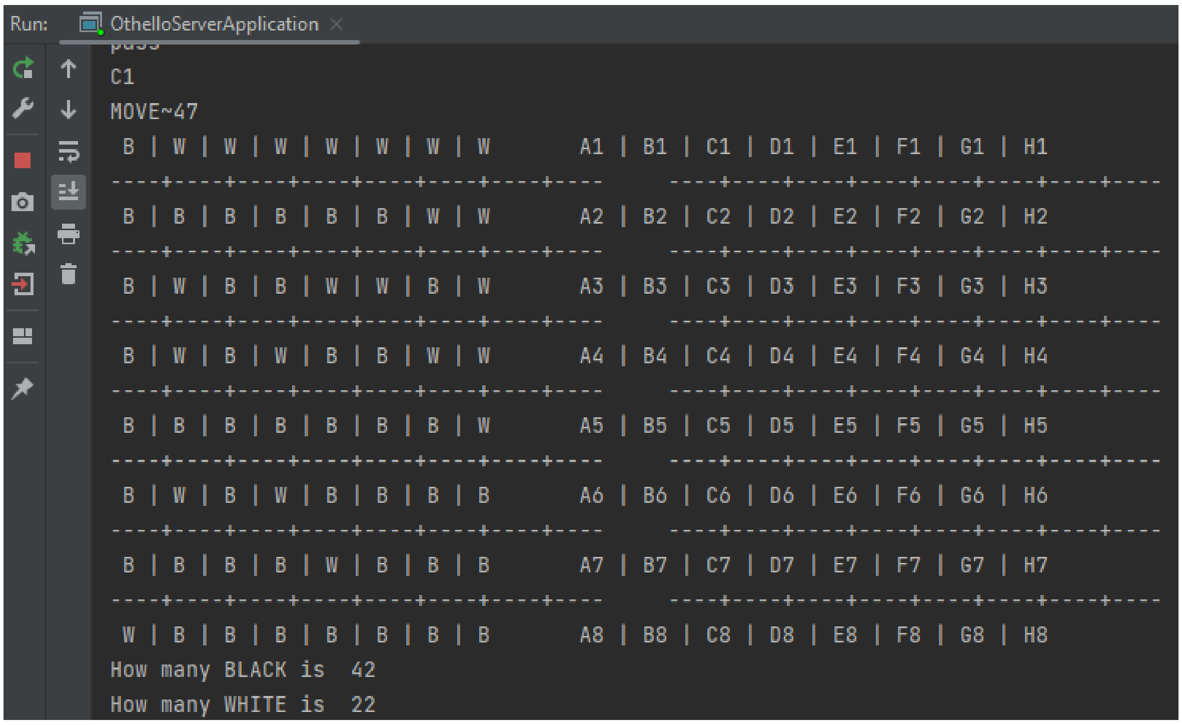
```
NEWGAME~C1~C2
NEWGAME~C1~C2
MOVE~26
    |    |    |    |    |    |    |         A1
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A2
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A3
----+----+----+----+----+----+----+----
    |    | B  | B  | B  |    |    |         A4
----+----+----+----+----+----+----+----
    |    |    | B  | W  |    |    |         A5
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A6
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A7
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A8
```

4. Play until the game is finished.



```
 B | W | W | W | W | W | W | W        A1 | B1 | C1 |
----+----+----+----+----+----+----+----      ----+--
 B | B | B | B | B | B | W | W        A2 | B2 | C2 |
----+----+----+----+----+----+----+----      ----+--
 B | W | B | B | W | W | B | W        A3 | B3 | C3 |
----+----+----+----+----+----+----+----      ----+--
 B | W | B | W | B | B | W | W        A4 | B4 | C4 |
----+----+----+----+----+----+----+----      ----+--
 B | B | B | B | B | B | B | W        A5 | B5 | C5 |
----+----+----+----+----+----+----+----      ----+--
 B | W | B | W | B | B | B | B        A6 | B6 | C6 |
----+----+----+----+----+----+----+----      ----+--
 B | B | B | B | W | B | B | B        A7 | B7 | C7 |
----+----+----+----+----+----+----+----      ----+--
 W | B | B | B | B | B | B | B        A8 | B8 | C8 |
INCOMING MOVE~47
Game is over because VICTORY by C1
Play as Human(1) or Computer(2)?
New Game has started with C1 and C2
```

```
 B | B | B | B | B | B | W | W        A2 | B2
----+----+----+----+----+----+----+----
 B | W | B | B | W | W | B | W        A3 | B3
----+----+----+----+----+----+----+----
 B | W | B | W | B | B | W | W        A4 | B4
----+----+----+----+----+----+----+----
 B | B | B | B | B | B | B | W        A5 | B5
----+----+----+----+----+----+----+----
 B | W | B | W | B | B | B | B        A6 | B6
----+----+----+----+----+----+----+----
 B | B | B | B | W | B | B | B        A7 | B7
----+----+----+----+----+----+----+----
 W | B | B | B | B | B | B | B        A8 | B8
INCOMING MOVE~47
pass
OUTGOING MOVE~64
Game is over because VICTORY by C1
Play as Human(1) or Computer(2)?
New Game has started with C1 and C2
```



```
pass
C1
MOVE~47
 B | W | W | W | W | W | W | W        A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 B | B | B | B | B | B | W | W        A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 B | W | B | B | W | W | B | W        A3 | B3 | C3 | D3 | E3 | F3 | G3 | H3
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 B | W | B | W | B | B | W | W        A4 | B4 | C4 | D4 | E4 | F4 | G4 | H4
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 B | B | B | B | B | B | B | W        A5 | B5 | C5 | D5 | E5 | F5 | G5 | H5
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 B | W | B | W | B | B | B | B        A6 | B6 | C6 | D6 | E6 | F6 | G6 | H6
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 B | B | B | B | W | B | B | B        A7 | B7 | C7 | D7 | E7 | F7 | G7 | H7
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 W | B | B | B | B | B | B | B        A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
How many BLACK is  42
How many WHITE is  22
```

Testing Report for FR4

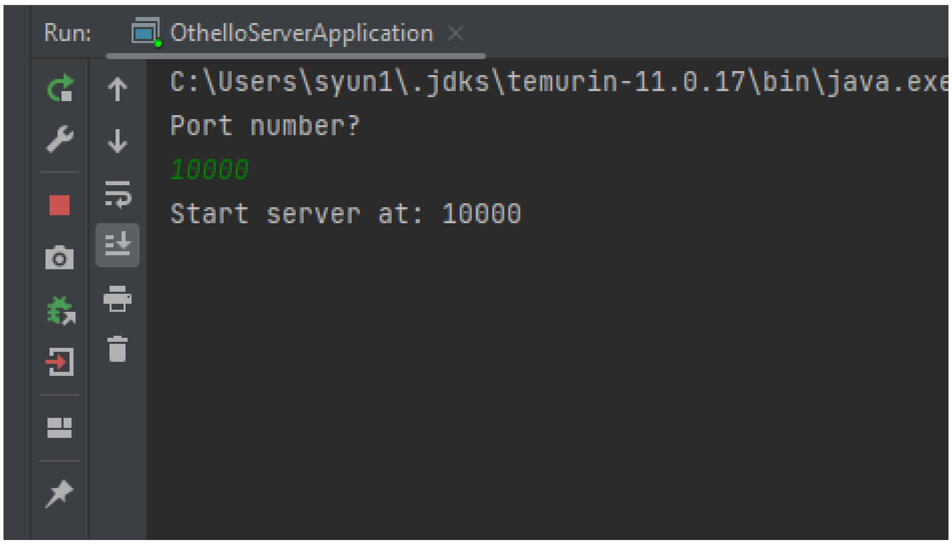
FR4: Setup for the server

Expected behavior:

- When the server is started, it will ask the user to input a port number where it will accept connections. If this number is already in use, the server will ask again.
- If the other server uses a specific port number, the server can not access the port number and the UI will ask the port number to user again.

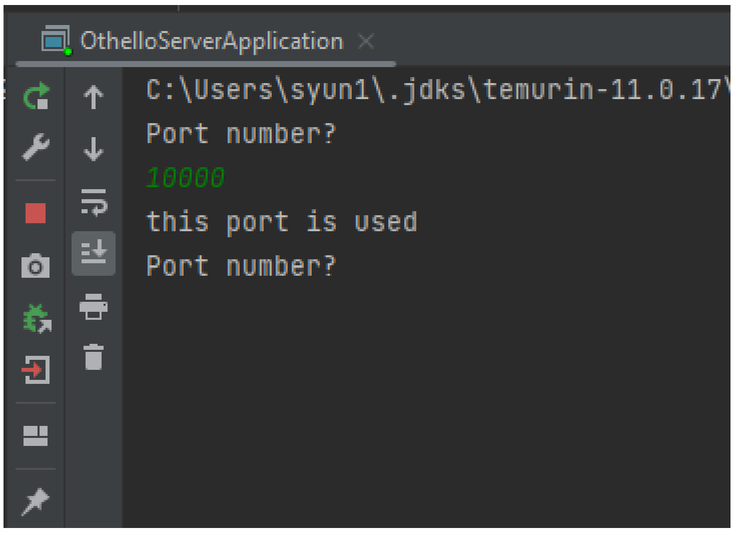
Testing result

1. The server uses port number 10000.



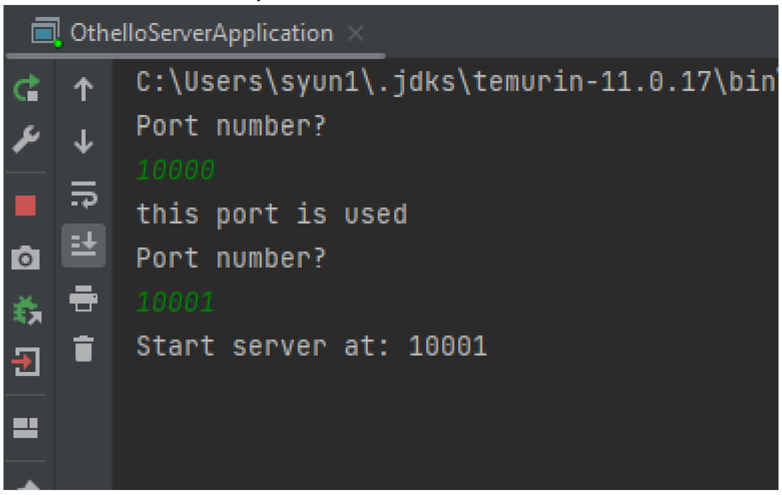
```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
Port number?
10000
Start server at: 10000
```

2. Another server tries to access 10000.



```
C:\Users\syun1\.jdks\temurin-11.0.17
Port number?
10000
this port is used
Port number?
```

3. Put another port number which is accessible.



```
C:\Users\syun1\.jdks\temurin-11.0.17\bin
Port number?
10000
this port is used
Port number?
10001
Start server at: 10001
```

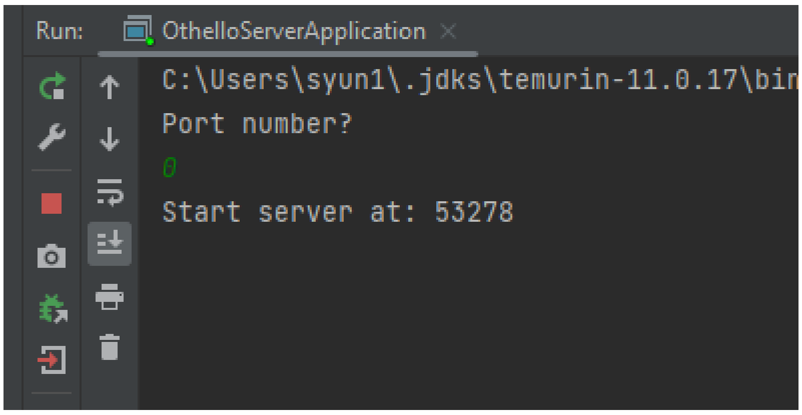
Testing Report for FR5
FR5: Setup for the client

Expected behavior:

- When the client is started, it should ask the user for the IP-address and port number of the server to connect to.
- Every case where the IP address is not correct and the IP address is correct but the port number is not correct, the UI should ask the user again.

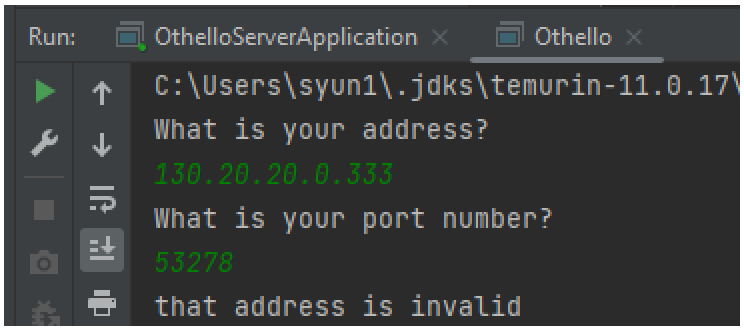
Testing result

1. The server starts.



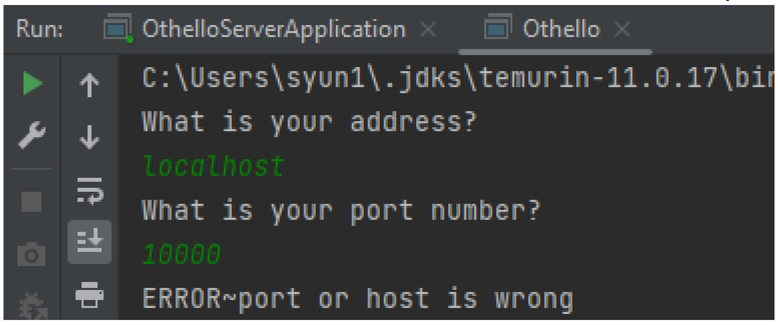
```
C:\Users\syun1\.jdks\temurin-11.0.17\bin
Port number?
0
Start server at: 53278
```

2. The client uses an incorrect IP address and the correct port.



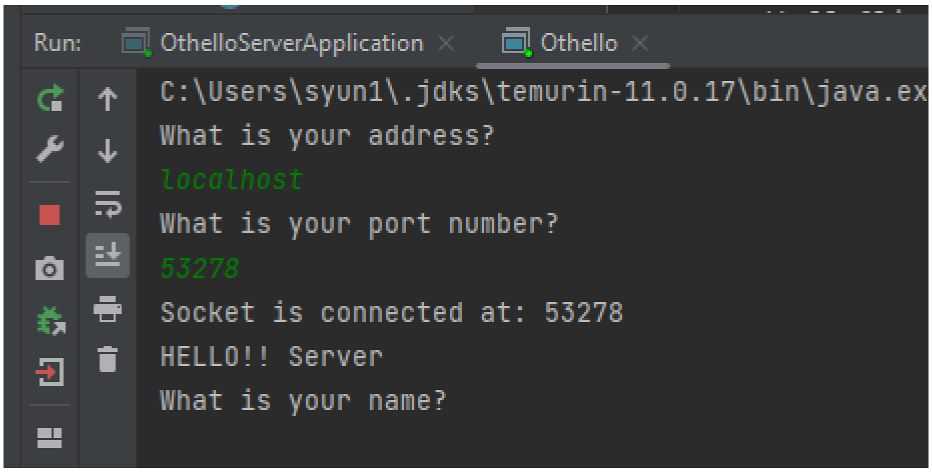
```
C:\Users\syun1\.jdks\temurin-11.0.17\
What is your address?
130.20.20.0.333
What is your port number?
53278
that address is invalid
```

3. The client uses correct IP address and incorrect port number.



```
C:\Users\syun1\.jdks\temurin-11.0.17\bin
What is your address?
localhost
What is your port number?
10000
ERROR~port or host is wrong
```

4. The client uses both correct input.



```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.ex
What is your address?
localhost
What is your port number?
53278
Socket is connected at: 53278
HELLO!! Server
What is your name?
```

Testing Report for FR6

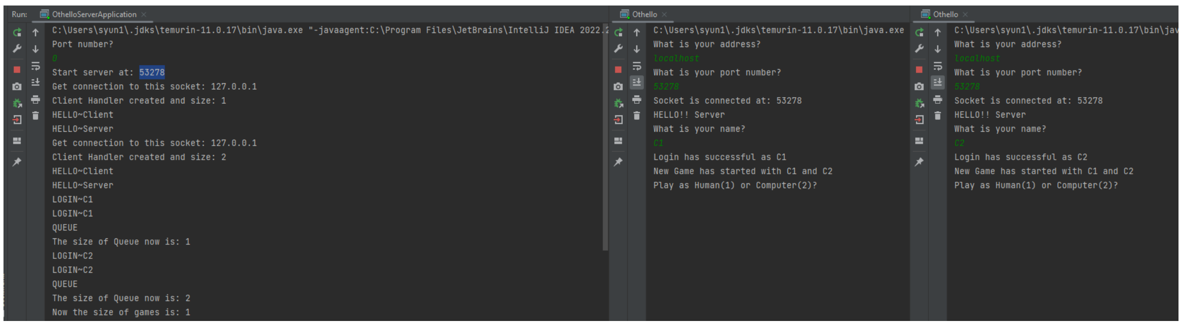
FR6: Hint for human

Expected behavior:

- When the client is controlled by a human player, the user can request a possible valid move as a hint via the TUI.

Testing result

1. Both the server and two clients start.

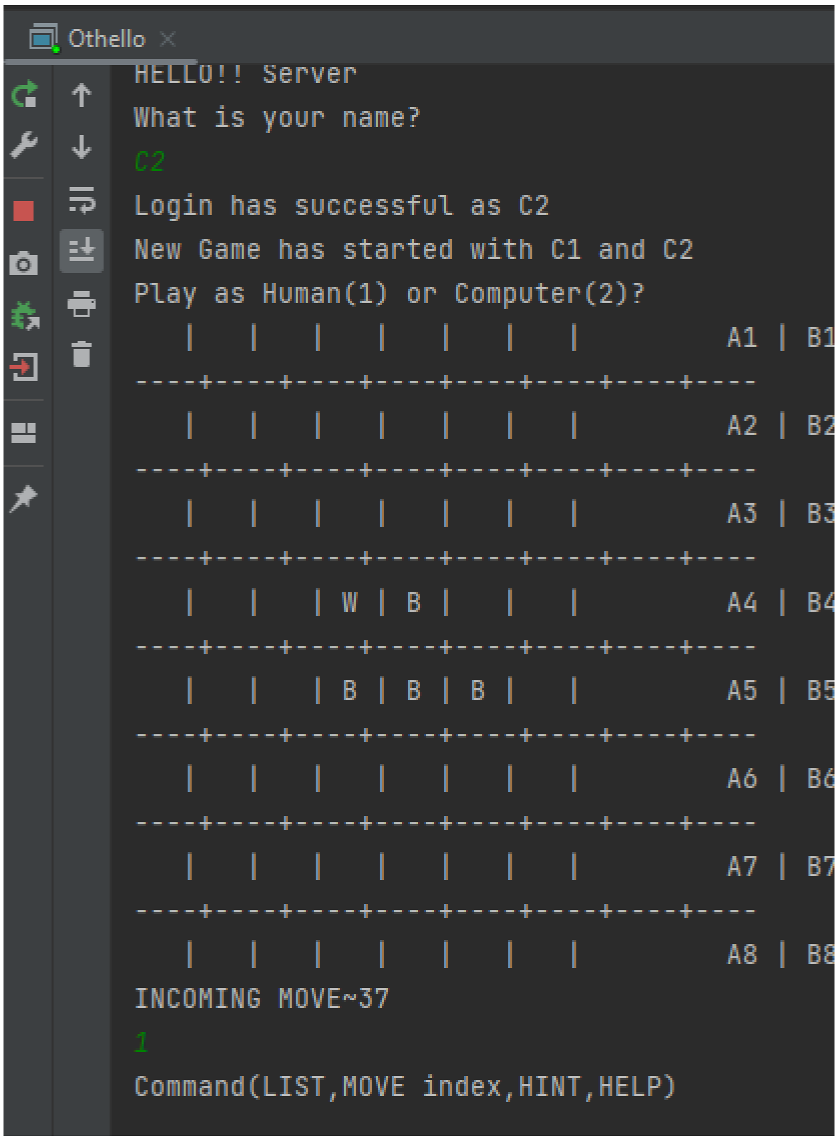


```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2
Port number?
0
Start server at: 53278
Get connection to this socket: 127.0.0.1
Client Handler created and size: 1
HELLO~Client
HELLO~Server
Get connection to this socket: 127.0.0.1
Client Handler created and size: 2
HELLO~Client
HELLO~Server
LOGIN~C1
LOGIN~C1
QUEUE
The size of Queue now is: 1
LOGIN~C2
LOGIN~C2
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
```

```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
What is your address?
localhost
What is your port number?
53278
Socket is connected at: 53278
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

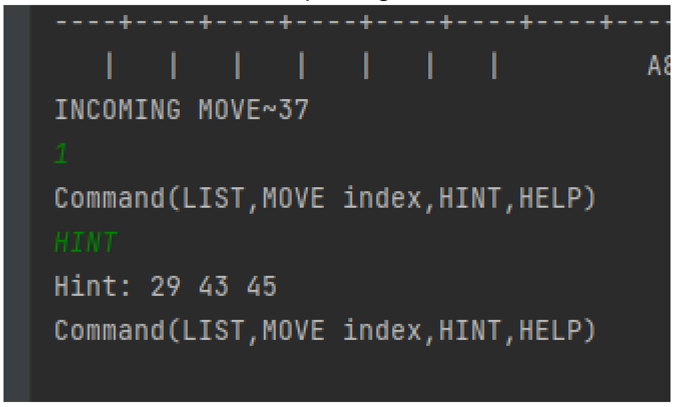
```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\jav
What is your address?
localhost
What is your port number?
53278
Socket is connected at: 53278
HELLO!! Server
What is your name?
C2
Login has successful as C2
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

2. One of the clients selects a Human player.



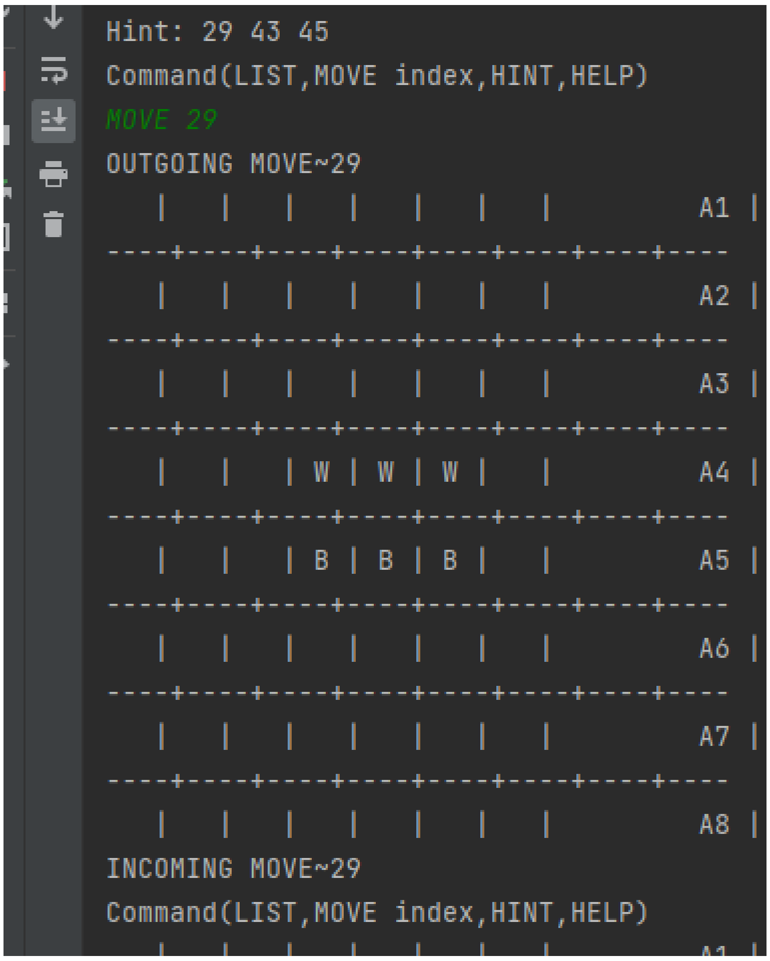
```
HELLO!! Server
What is your name?
C2
Login has successful as C2
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
    |   |   |   |   |   |   |         A1 | B1
----+---+---+---+---+---+---+----
    |   |   |   |   |   |   |         A2 | B2
----+---+---+---+---+---+---+----
    |   |   |   |   |   |   |         A3 | B3
----+---+---+---+---+---+---+----
    |   |   | W | B |   |   |         A4 | B4
----+---+---+---+---+---+---+----
    |   |   | B | B | B |   |         A5 | B5
----+---+---+---+---+---+---+----
    |   |   |   |   |   |   |         A6 | B6
----+---+---+---+---+---+---+----
    |   |   |   |   |   |   |         A7 | B7
----+---+---+---+---+---+---+----
    |   |   |   |   |   |   |         A8 | B8
INCOMING MOVE~37
1
Command(LIST,MOVE index,HINT,HELP)
```

3. Ask Hint before putting the move as command.



```
----+---+---+---+---+---+---+---
    |   |   |   |   |   |   |         A8
INCOMING MOVE~37
1
Command(LIST,MOVE index,HINT,HELP)
HINT
Hint: 29 43 45
Command(LIST,MOVE index,HINT,HELP)
```

4. Move commands are successful using hint numbers.



```
Hint: 29 43 45
Command(LIST,MOVE index,HINT,HELP)
MOVE 29
OUTGOING MOVE~29
   |   |   |   |   |   |   |       A1 |
----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |       A2 |
----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |       A3 |
----+----+----+----+----+----+----+----
   |   |   | W | W | W |   |       A4 |
----+----+----+----+----+----+----+----
   |   |   | B | B | B |   |       A5 |
----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |       A6 |
----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |       A7 |
----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |       A8 |
INCOMING MOVE~29
Command(LIST,MOVE index,HINT,HELP)
```

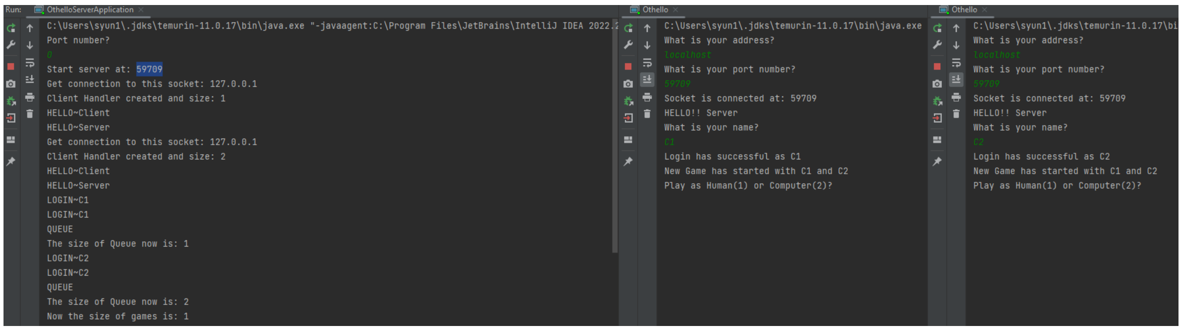
Testing Report for FR7

FR7: Auto play as AI at client

Expected behavior:
- The client can play a full game automatically as the AI without intervention by the user.
- During the game, the UI ignores user input.

Testing result

1. Both the server and two clients start.

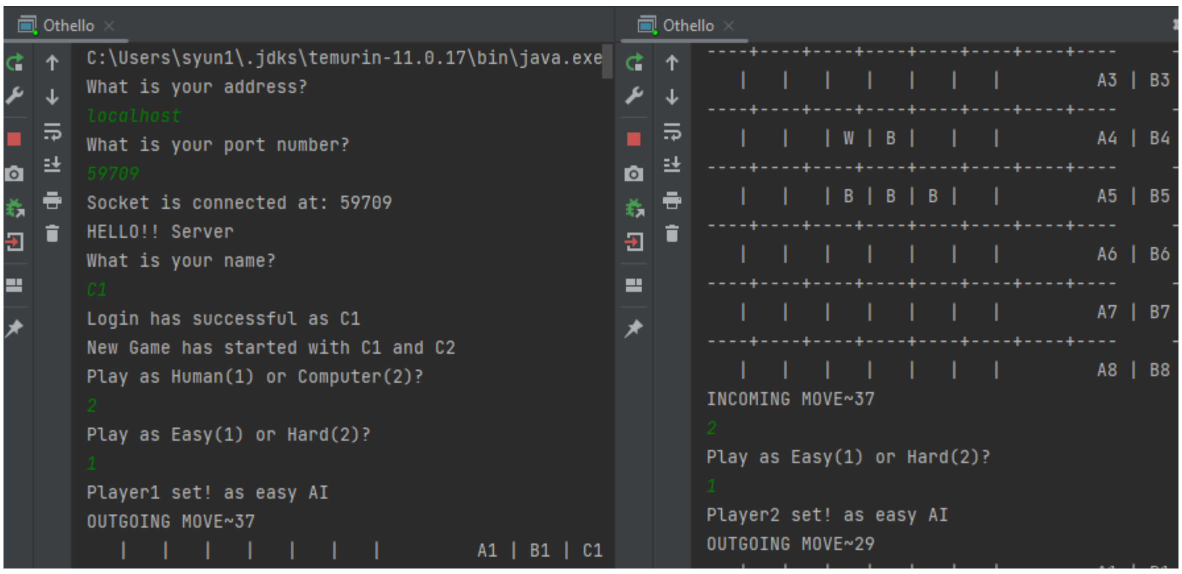


```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2
Port number?
0
Start server at: 59709
Get connection to this socket: 127.0.0.1
Client Handler created and size: 1
HELLO~Client
HELLO~Server
Get connection to this socket: 127.0.0.1
Client Handler created and size: 2
HELLO~Client
HELLO~Server
LOGIN~C1
LOGIN~C1
QUEUE
The size of Queue now is: 1
LOGIN~C2
LOGIN~C2
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
```

```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
What is your address?
localhost
What is your port number?
59709
Socket is connected at: 59709
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

```
C:\Users\syun1\.jdks\temurin-11.0.17\bi
What is your address?
localhost
What is your port number?
59709
Socket is connected at: 59709
HELLO!! Server
What is your name?
C2
Login has successful as C2
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

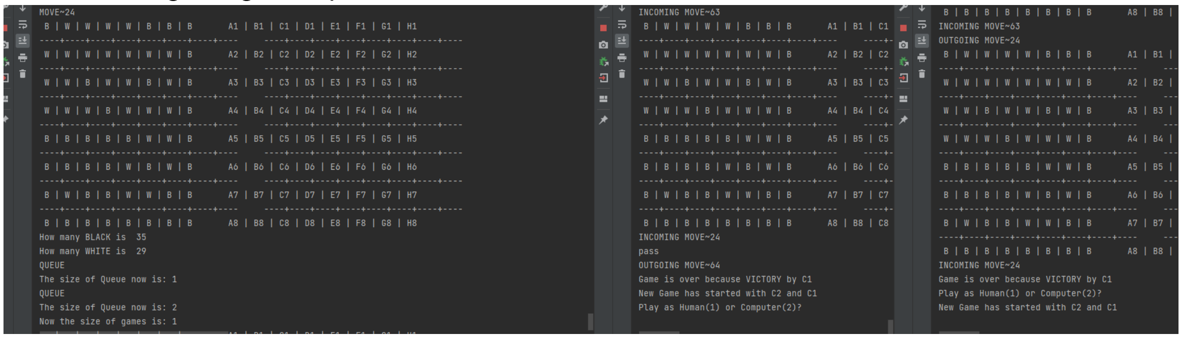
2. Both clients select Computer Player.



```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
What is your address?
localhost
What is your port number?
59709
Socket is connected at: 59709
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
2
Play as Easy(1) or Hard(2)?
1
Player1 set! as easy AI
OUTGOING MOVE~37
    |    |    |    |    |    |    |           A1 | B1 | C1
```

```
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |           A3 | B3
----+----+----+----+----+----+----+----
    |    |    | W  | B  |    |    |           A4 | B4
----+----+----+----+----+----+----+----
    |    |    | B  | B  | B  |    |           A5 | B5
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |           A6 | B6
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |           A7 | B7
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |           A8 | B8
INCOMING MOVE~37
2
Play as Easy(1) or Hard(2)?
1
Player2 set! as easy AI
OUTGOING MOVE~29
```

3. During the game, put some commands in the UI.



```
MOVE~24
 B | W | W | W | W | B | B | B       A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 W | W | W | W | W | W | W | B       A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 W | W | B | W | W | B | W | B       A3 | B3 | C3 | D3 | E3 | F3 | G3 | H3
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 W | W | W | B | W | W | W | B       A4 | B4 | C4 | D4 | E4 | F4 | G4 | H4
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 B | B | B | B | B | W | W | B       A5 | B5 | C5 | D5 | E5 | F5 | G5 | H5
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 B | B | B | B | W | B | W | B       A6 | B6 | C6 | D6 | E6 | F6 | G6 | H6
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 B | W | B | B | W | W | B | B       A7 | B7 | C7 | D7 | E7 | F7 | G7 | H7
----+----+----+----+----+----+----+----     ----+----+----+----+----+----+----+----
 B | B | B | B | B | B | B | B       A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
How many BLACK is  35
How many WHITE is  29
QUEUE
The size of Queue now is: 1
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
```

```
INCOMING MOVE~63
 B | W | W | W | W | B | B | B       A1 | B1 | C1
----+----+----+----+----+----+----+----     ----+-
 W | W | W | W | W | W | W | B       A2 | B2 | C2
----+----+----+----+----+----+----+----     ----+-
 W | W | B | W | W | B | W | B       A3 | B3 | C3
----+----+----+----+----+----+----+----     ----+-
 W | W | W | B | W | W | W | B       A4 | B4 | C4
----+----+----+----+----+----+----+----     ----+-
 B | B | B | B | B | W | W | B       A5 | B5 | C5
----+----+----+----+----+----+----+----     ----+-
 B | B | B | B | W | B | W | B       A6 | B6 | C6
----+----+----+----+----+----+----+----     ----+-
 B | W | B | B | W | W | B | B       A7 | B7 | C7
----+----+----+----+----+----+----+----     ----+-
 B | B | B | B | B | B | B | B       A8 | B8 | C8
INCOMING MOVE~24
pass
OUTGOING MOVE~64
Game is over because VICTORY by C1
New Game has started with C2 and C1
Play as Human(1) or Computer(2)?
```

```
 B | B | B | B | B | B | B | B       A8 | B8 |
INCOMING MOVE~63
OUTGOING MOVE~24
 B | W | W | W | W | B | B | B       A1 | B1 |
----+----+----+----+----+----+----+----     ---
 W | W | W | W | W | W | W | B       A2 | B2 |
----+----+----+----+----+----+----+----     ---
 W | W | B | W | W | B | W | B       A3 | B3 |
----+----+----+----+----+----+----+----     ---
 W | W | W | B | W | W | W | B       A4 | B4 |
----+----+----+----+----+----+----+----     ---
 B | B | B | B | B | W | W | B       A5 | B5 |
----+----+----+----+----+----+----+----     ---
 B | B | B | B | W | B | W | B       A6 | B6 |
----+----+----+----+----+----+----+----     ---
 B | W | B | B | W | W | B | B       A7 | B7 |
----+----+----+----+----+----+----+----     ---
 B | B | B | B | B | B | B | B       A8 | B8 |
INCOMING MOVE~24
Game is over because VICTORY by C1
Play as Human(1) or Computer(2)?
New Game has started with C2 and C1
```

Automatically finished.

Testing Report for FR8

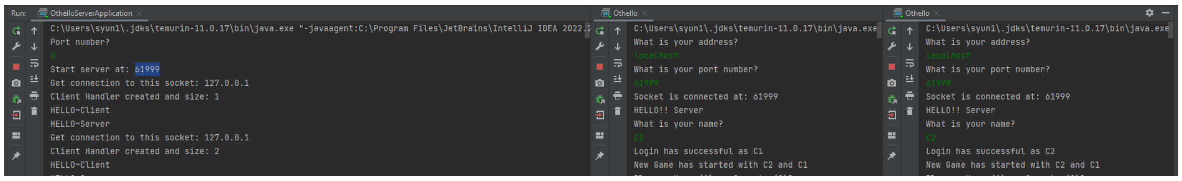
FR8: Adjust difficulty of AI at client

Expected behavior:

- The user can adjust the AI difficulty via the TUI.
- We can select AI difficulty when the game is starting.

Testing result

1. Both the server and two clients start.

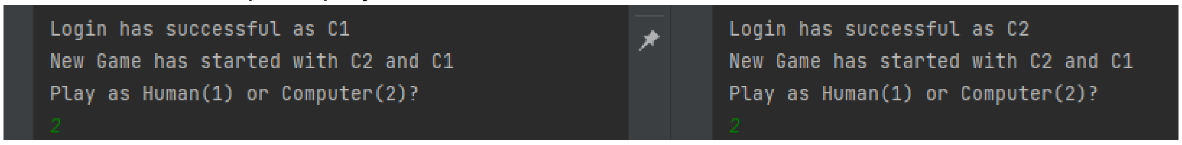


```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2
Port number?
0
Start server at: 61999
Get connection to this socket: 127.0.0.1
Client Handler created and size: 1
HELLO~Client
HELLO~Server
Get connection to this socket: 127.0.0.1
Client Handler created and size: 2
HELLO~Client
```

```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
What is your address?
localhost
What is your port number?
61999
Socket is connected at: 61999
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C2 and C1
```

```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
What is your address?
localhost
What is your port number?
61999
Socket is connected at: 61999
HELLO!! Server
What is your name?
C2
Login has successful as C2
New Game has started with C2 and C1
```

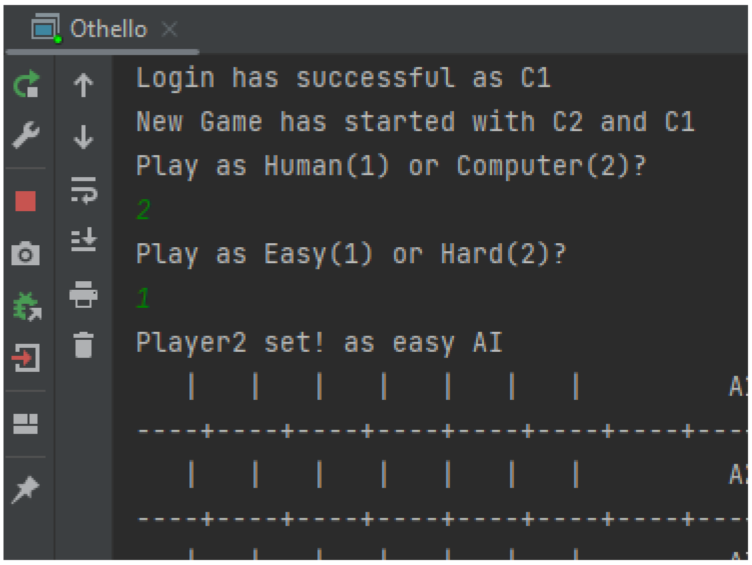
2. Select Computer player.



```
Login has successful as C1
New Game has started with C2 and C1
Play as Human(1) or Computer(2)?
2
```

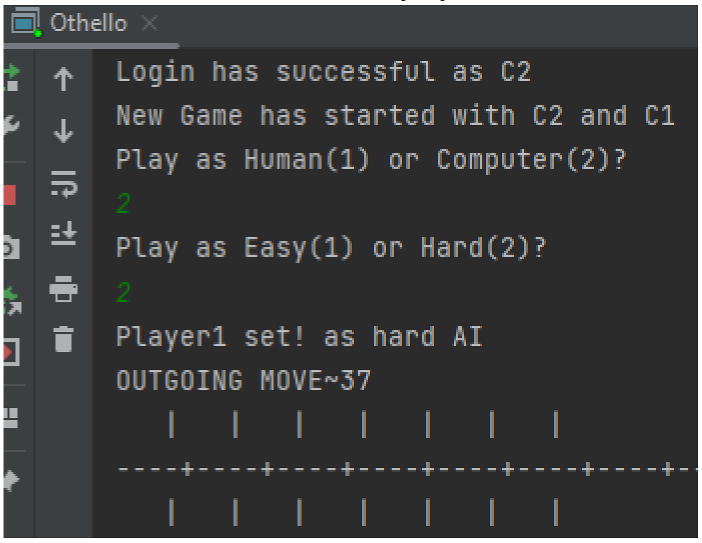
```
Login has successful as C2
New Game has started with C2 and C1
Play as Human(1) or Computer(2)?
2
```

3. Select one of the difficulties by one of the clients.



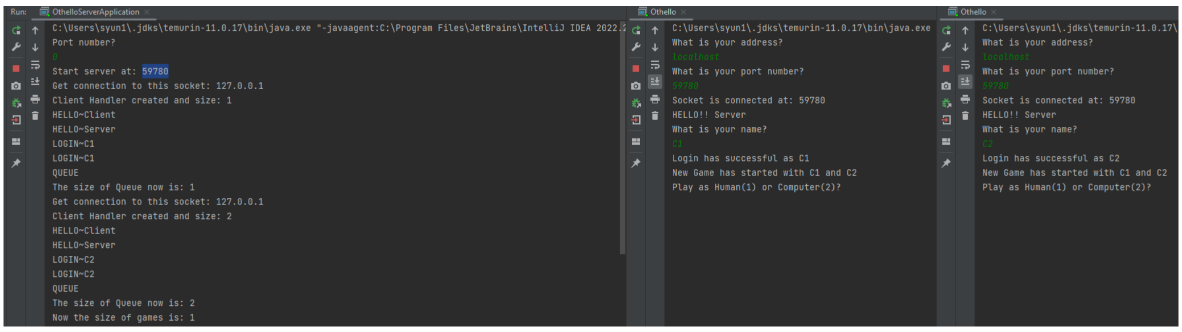
```
Login has successful as C1
New Game has started with C2 and C1
Play as Human(1) or Computer(2)?
2
Play as Easy(1) or Hard(2)?
1
Player2 set! as easy AI
   |   |   |   |   |   |   |         A
----+----+----+----+----+----+----+---
   |   |   |   |   |   |   |         A
----+----+----+----+----+----+----+---
```

4. Select the other difficulty by the other client.



```
Login has successful as C2
New Game has started with C2 and C1
Play as Human(1) or Computer(2)?
2
Play as Easy(1) or Hard(2)?
2
Player1 set! as hard AI
OUTGOING MOVE~37
   |   |   |   |   |   |   |
----+----+----+----+----+----+----+-
   |   |   |   |   |   |   |
```

Testing Report for FR9
FR9: Continue to new game
Expected behavior: - Whenever a game has finished (except when the server is disconnected), a new game can be played without needing to establish a new connection in between. - Soon after the game playing, clients will be in the queue - One of the clients has disconnected, the other client will be in the queue immediately.
Testing result 1. Both the server and two clients start.

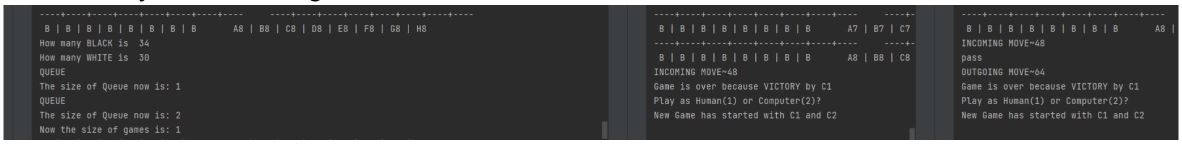


```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2
Port number?
0
Start server at: 59780
Get connection to this socket: 127.0.0.1
Client Handler created and size: 1
HELLO~Client
HELLO~Server
LOGIN~C1
LOGIN~C1
QUEUE
The size of Queue now is: 1
Get connection to this socket: 127.0.0.1
Client Handler created and size: 2
HELLO~Client
HELLO~Server
LOGIN~C2
LOGIN~C2
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
```

```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
What is your address?
localhost
What is your port number?
59780
Socket is connected at: 59780
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

```
C:\Users\syun1\.jdks\temurin-11.0.17\
What is your address?
localhost
What is your port number?
59780
Socket is connected at: 59780
HELLO!! Server
What is your name?
C2
Login has successful as C2
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

2. Play a standard game and finish.

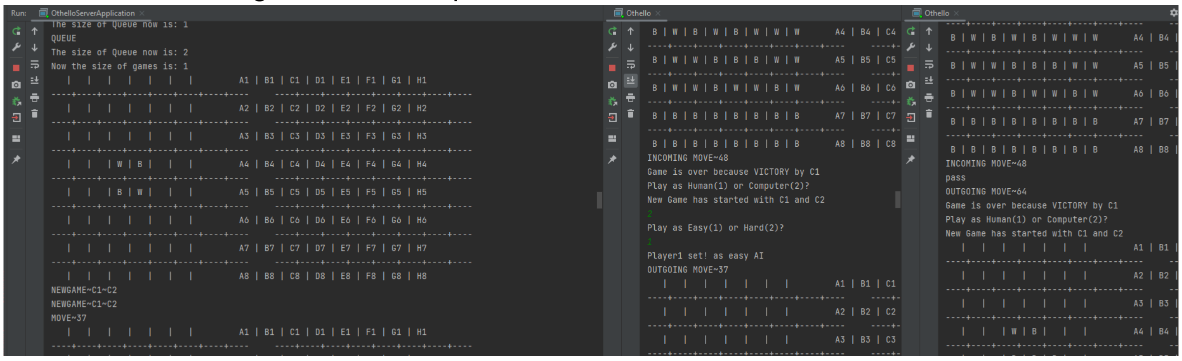


```
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
 B | B | B | B | B | B | B | B       A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
How many BLACK is  34
How many WHITE is  30
QUEUE
The size of Queue now is: 1
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
```

```
----+----+----+----+----+----+----+----      ----+-
 B | B | B | B | B | B | B | B       A7 | B7 | C7
----+----+----+----+----+----+----+----      ----+-
 B | B | B | B | B | B | B | B       A8 | B8 | C8
INCOMING MOVE~48
Game is over because VICTORY by C1
Play as Human(1) or Computer(2)?
New Game has started with C1 and C2
```

```
----+----+----+----+----+----+----+----
 B | B | B | B | B | B | B | B       A8 |
INCOMING MOVE~48
pass
OUTGOING MOVE~64
Game is over because VICTORY by C1
Play as Human(1) or Computer(2)?
New Game has started with C1 and C2
```

3. Start a new game in consequence.

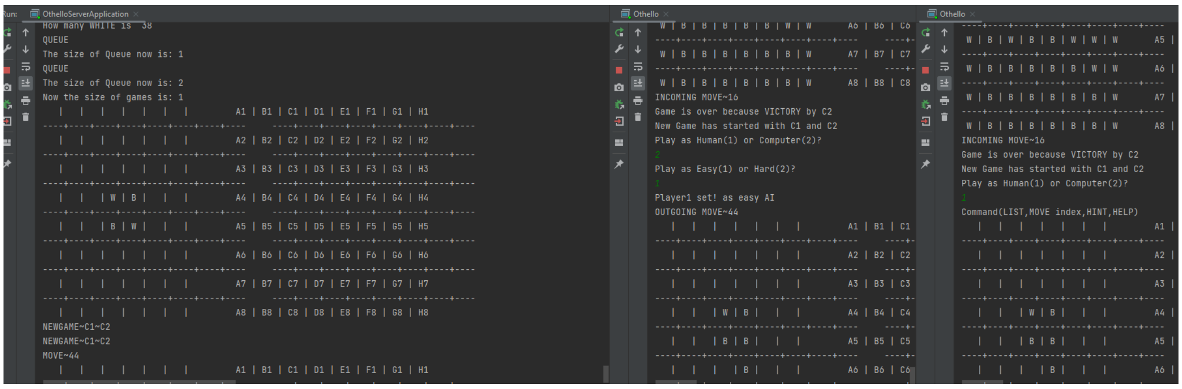


```
The size of Queue now is: 1
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
   |   |   |   |   |   |   |         A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A3 | B3 | C3 | D3 | E3 | F3 | G3 | H3
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   | W | B |   |   |         A4 | B4 | C4 | D4 | E4 | F4 | G4 | H4
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   | B | W |   |   |         A5 | B5 | C5 | D5 | E5 | F5 | G5 | H5
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A6 | B6 | C6 | D6 | E6 | F6 | G6 | H6
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A7 | B7 | C7 | D7 | E7 | F7 | G7 | H7
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
NEWGAME~C1~C2
NEWGAME~C1~C2
MOVE~37
   |   |   |   |   |   |   |         A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
```

```
 B | W | B | W | B | W | W | W       A4 | B4 | C4
----+----+----+----+----+----+----+----      ----+-
 B | W | W | B | B | B | W | W       A5 | B5 | C5
----+----+----+----+----+----+----+----      ----+-
 B | W | W | B | W | W | B | W       A6 | B6 | C6
----+----+----+----+----+----+----+----      ----+-
 B | B | B | B | B | B | B | B       A7 | B7 | C7
----+----+----+----+----+----+----+----      ----+-
 B | B | B | B | B | B | B | B       A8 | B8 | C8
INCOMING MOVE~48
Game is over because VICTORY by C1
Play as Human(1) or Computer(2)?
New Game has started with C1 and C2
2
Play as Easy(1) or Hard(2)?
1
Player1 set! as easy AI
OUTGOING MOVE~37
   |   |   |   |   |   |   |         A1 | B1 | C1
----+----+----+----+----+----+----+----      ----+-
   |   |   |   |   |   |   |         A2 | B2 | C2
----+----+----+----+----+----+----+----      ----+-
   |   |   |   |   |   |   |         A3 | B3 | C3
----+----+----+----+----+----+----+----      ----+-
```

```
----+----+----+----+----+----+----+----      --
 B | W | B | W | B | W | W | W       A4 | B4 |
----+----+----+----+----+----+----+----      --
 B | W | W | B | B | B | W | W       A5 | B5 |
----+----+----+----+----+----+----+----      --
 B | W | W | B | W | W | B | W       A6 | B6 |
----+----+----+----+----+----+----+----      --
 B | B | B | B | B | B | B | B       A7 | B7 |
----+----+----+----+----+----+----+----      --
 B | B | B | B | B | B | B | B       A8 | B8 |
INCOMING MOVE~48
pass
OUTGOING MOVE~64
Game is over because VICTORY by C1
Play as Human(1) or Computer(2)?
New Game has started with C1 and C2
   |   |   |   |   |   |   |         A1 | B1 |
----+----+----+----+----+----+----+----      --
   |   |   |   |   |   |   |         A2 | B2 |
----+----+----+----+----+----+----+----      --
   |   |   |   |   |   |   |         A3 | B3 |
----+----+----+----+----+----+----+----      --
   |   |   | W | B |   |   |         A4 | B4 |
----+----+----+----+----+----+----+----      --
```

4. Make one of the clients disconnected.

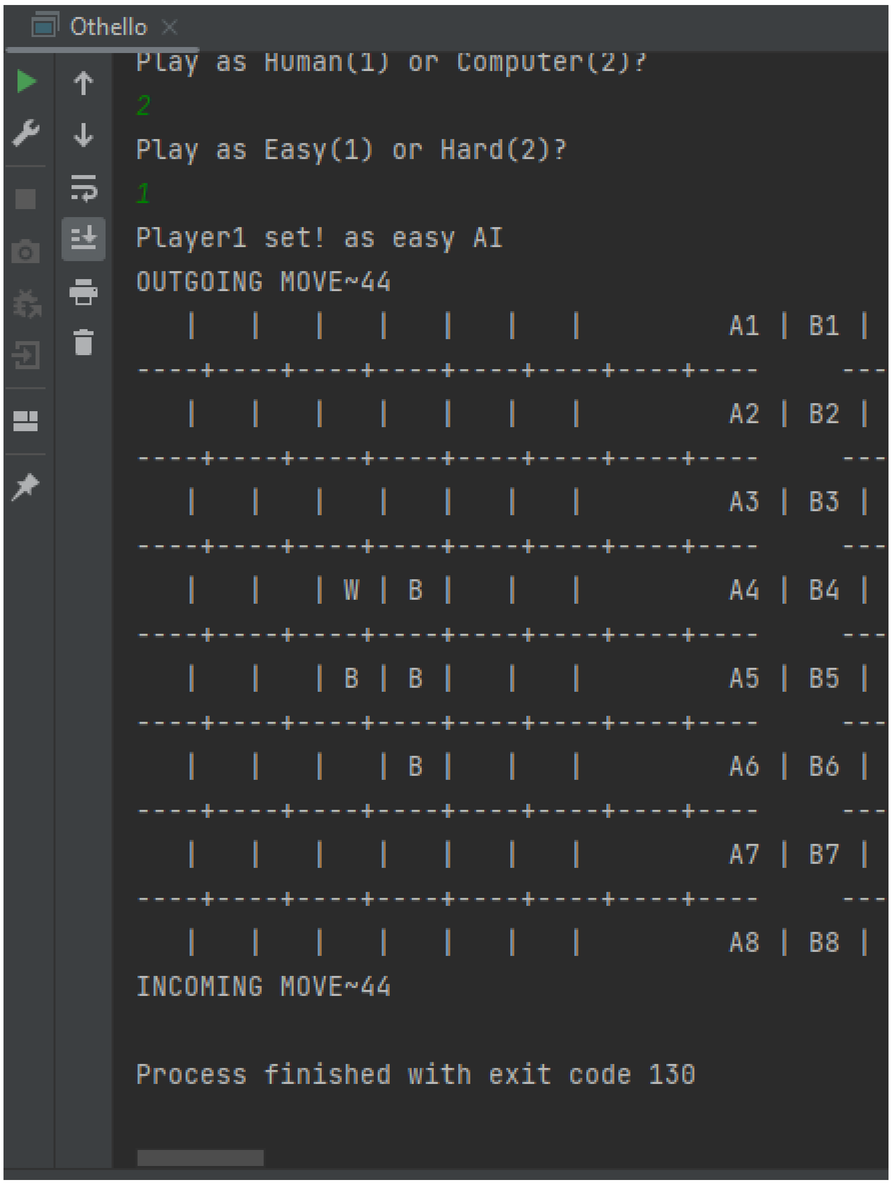


```
How many WHITE is  38
QUEUE
The size of Queue now is: 1
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
   |   |   |   |   |   |   |         A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A3 | B3 | C3 | D3 | E3 | F3 | G3 | H3
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   | W | B |   |   |         A4 | B4 | C4 | D4 | E4 | F4 | G4 | H4
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   | B | W |   |   |         A5 | B5 | C5 | D5 | E5 | F5 | G5 | H5
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A6 | B6 | C6 | D6 | E6 | F6 | G6 | H6
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A7 | B7 | C7 | D7 | E7 | F7 | G7 | H7
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
NEWGAME~C1~C2
NEWGAME~C1~C2
MOVE~44
   |   |   |   |   |   |   |         A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
```

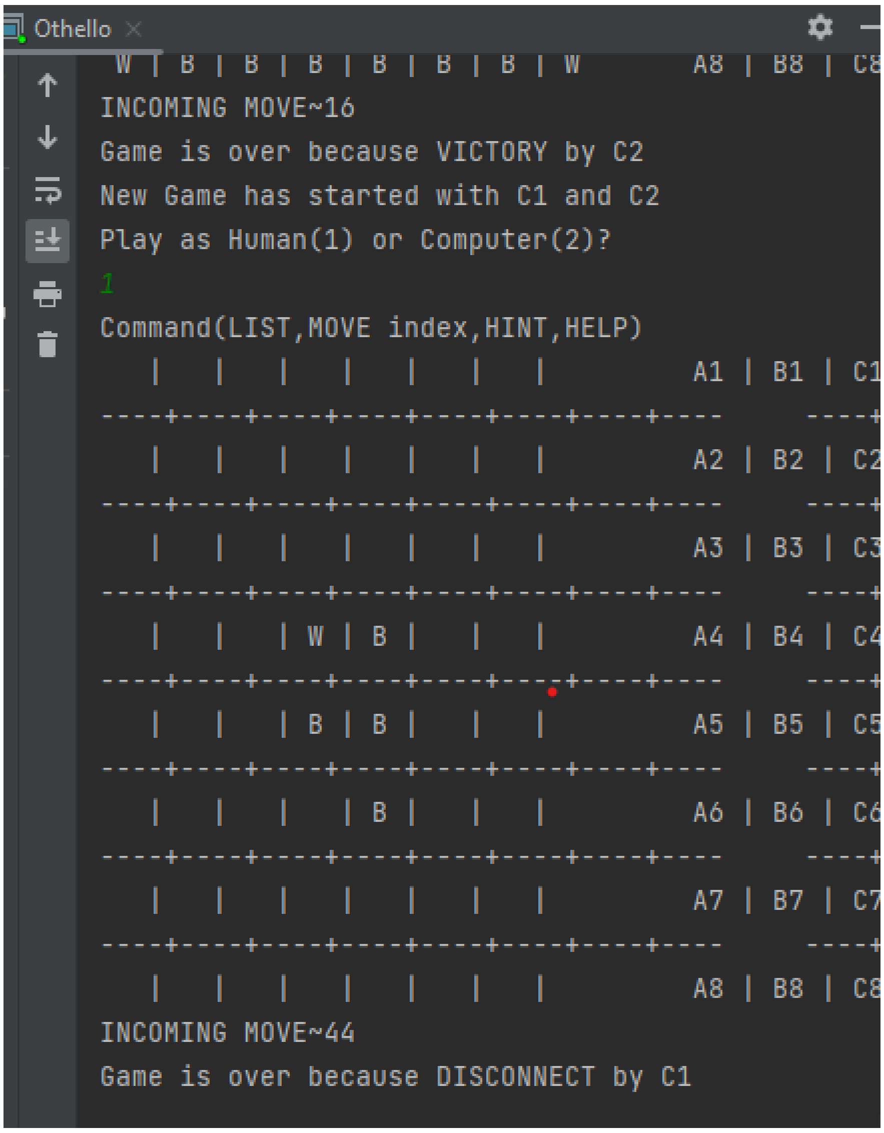
```
 W | B | B | B | B | B | W | W       A6 | B6 | C6
----+----+----+----+----+----+----+----      ----+-
 W | B | B | B | B | B | B | W       A7 | B7 | C7
----+----+----+----+----+----+----+----      ----+-
 W | B | B | B | B | B | B | W       A8 | B8 | C8
INCOMING MOVE~16
Game is over because VICTORY by C2
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
2
Play as Easy(1) or Hard(2)?
1
Player1 set! as easy AI
OUTGOING MOVE~44
   |   |   |   |   |   |   |         A1 | B1 | C1
----+----+----+----+----+----+----+----      ----+-
   |   |   |   |   |   |   |         A2 | B2 | C2
----+----+----+----+----+----+----+----      ----+-
   |   |   |   |   |   |   |         A3 | B3 | C3
----+----+----+----+----+----+----+----      ----+-
   |   |   | W | B |   |   |         A4 | B4 | C4
----+----+----+----+----+----+----+----      ----+-
   |   |   | B | B |   |   |         A5 | B5 | C5
----+----+----+----+----+----+----+----      ----+-
   |   |   |   | B |   |   |         A6 | B6 | C6
```

```
----+----+----+----+----+----+----+----
 W | B | W | B | B | W | W | W       A5 |
----+----+----+----+----+----+----+----
 W | B | B | B | B | B | W | W       A6 |
----+----+----+----+----+----+----+----
 W | B | B | B | B | B | B | W       A7 |
----+----+----+----+----+----+----+----
 W | B | B | B | B | B | B | W       A8 |
INCOMING MOVE~16
Game is over because VICTORY by C2
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
1
Command(LIST,MOVE index,HINT,HELP)
   |   |   |   |   |   |   |         A1 |
----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A2 |
----+----+----+----+----+----+----+----
   |   |   |   |   |   |   |         A3 |
----+----+----+----+----+----+----+----
   |   |   | W | B |   |   |         A4 |
----+----+----+----+----+----+----+----
   |   |   | B | B |   |   |         A5 |
----+----+----+----+----+----+----+----
   |   |   |   | B |   |   |         A6 |
```

New Game has started.



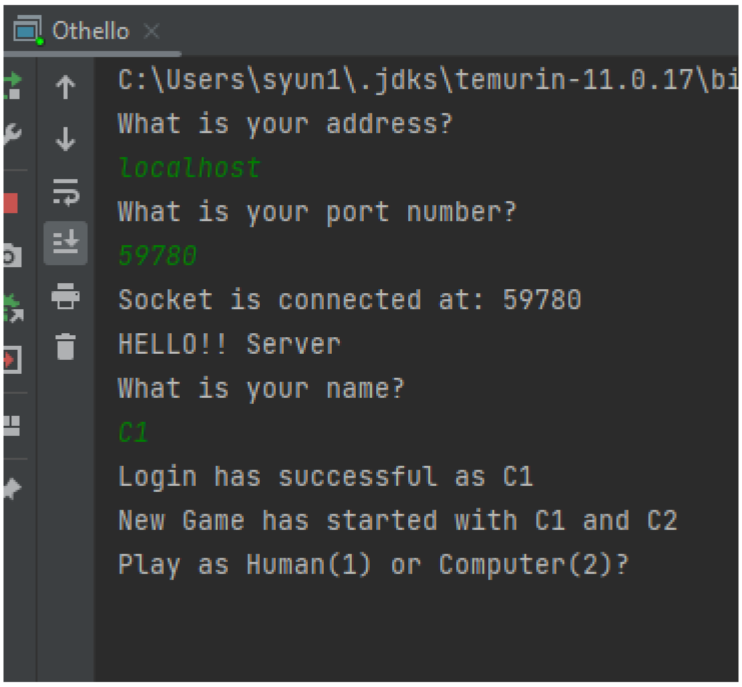
```
Play as Human(1) or Computer(2)?
2
Play as Easy(1) or Hard(2)?
1
Player1 set! as easy AI
OUTGOING MOVE~44
   |   |   |   |   |   |   |         A1 | B1 |
----+----+----+----+----+----+----+----     ---
   |   |   |   |   |   |   |         A2 | B2 |
----+----+----+----+----+----+----+----     ---
   |   |   |   |   |   |   |         A3 | B3 |
----+----+----+----+----+----+----+----     ---
   |   |   | W | B |   |   |         A4 | B4 |
----+----+----+----+----+----+----+----     ---
   |   |   | B | B |   |   |         A5 | B5 |
----+----+----+----+----+----+----+----     ---
   |   |   |   | B |   |   |         A6 | B6 |
----+----+----+----+----+----+----+----     ---
   |   |   |   |   |   |   |         A7 | B7 |
----+----+----+----+----+----+----+----     ---
   |   |   |   |   |   |   |         A8 | B8 |
INCOMING MOVE~44

Process finished with exit code 130
```



```
 W | B | B | B | B | B | B | W        A8 | B8 | C8
INCOMING MOVE~16
Game is over because VICTORY by C2
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
1
Command(LIST,MOVE index,HINT,HELP)
   |   |   |   |   |   |   |         A1 | B1 | C1
----+----+----+----+----+----+----+----      ----+
   |   |   |   |   |   |   |         A2 | B2 | C2
----+----+----+----+----+----+----+----      ----+
   |   |   |   |   |   |   |         A3 | B3 | C3
----+----+----+----+----+----+----+----      ----+
   |   |   | W | B |   |   |         A4 | B4 | C4
----+----+----+----+----+----+----+----      ----+
   |   |   | B | B |   |   |         A5 | B5 | C5
----+----+----+----+----+----+----+----      ----+
   |   |   |   | B |   |   |         A6 | B6 | C6
----+----+----+----+----+----+----+----      ----+
   |   |   |   |   |   |   |         A7 | B7 | C7
----+----+----+----+----+----+----+----      ----+
   |   |   |   |   |   |   |         A8 | B8 | C8
INCOMING MOVE~44
Game is over because DISCONNECT by C1
```

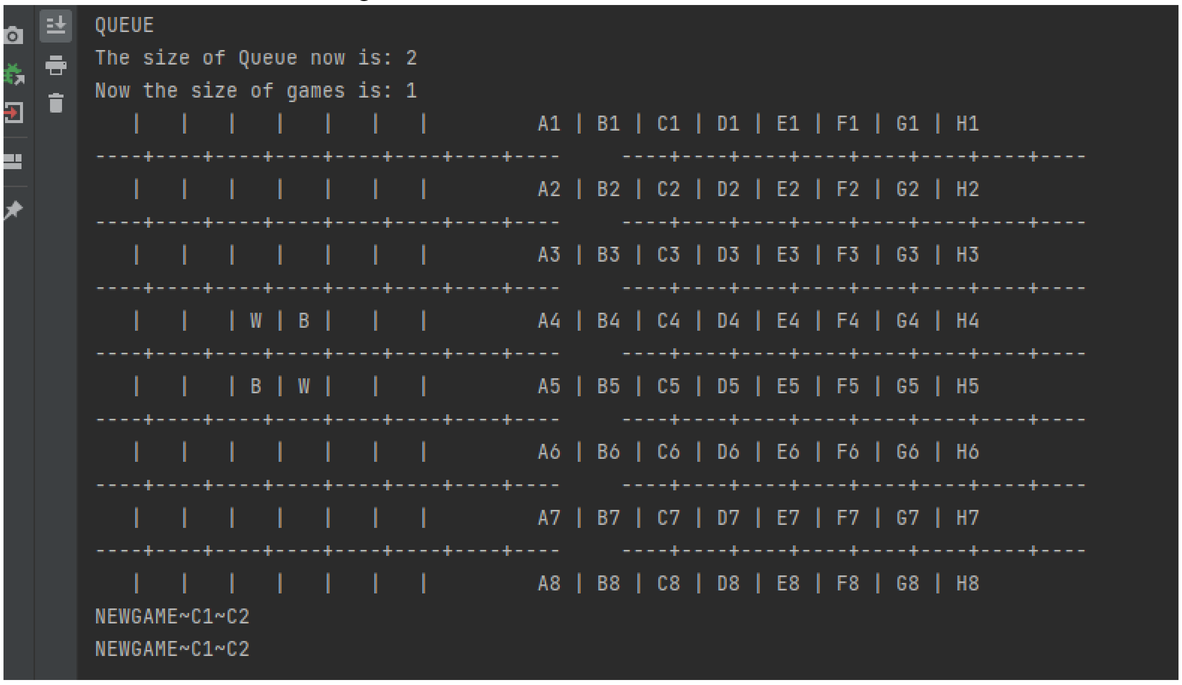
The opponent is disconnected. The other client is still connected and automatically in a queue.

5. If the other Client will be in queue, New game will start.



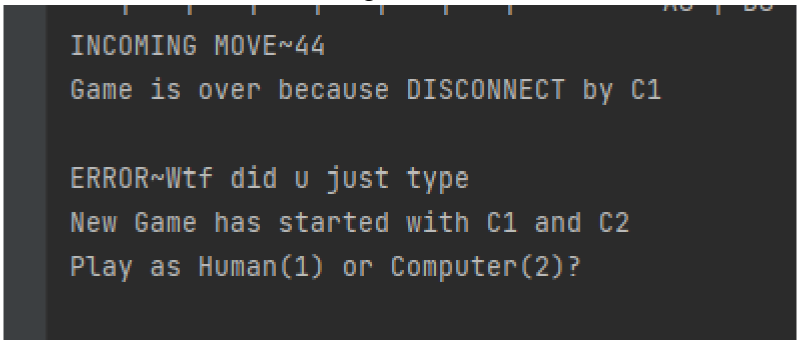
```
C:\Users\syun1\.jdks\temurin-11.0.17\bi
What is your address?
localhost
What is your port number?
59780
Socket is connected at: 59780
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

The other client accesses again.



```
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
    |    |    |    |    |    |    |            A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |            A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |            A3 | B3 | C3 | D3 | E3 | F3 | G3 | H3
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    | W  | B  |    |    |            A4 | B4 | C4 | D4 | E4 | F4 | G4 | H4
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    | B  | W  |    |    |            A5 | B5 | C5 | D5 | E5 | F5 | G5 | H5
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |            A6 | B6 | C6 | D6 | E6 | F6 | G6 | H6
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |            A7 | B7 | C7 | D7 | E7 | F7 | G7 | H7
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |            A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
NEWGAME~C1~C2
NEWGAME~C1~C2
```

The server initializes a new game.



```
INCOMING MOVE~44
Game is over because DISCONNECT by C1

ERROR~Wtf did u just type
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
```

The client can start a new game without reconnecting to the socket.

Testing Report for FR10

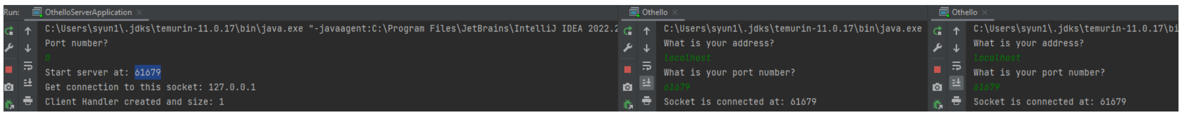
FR10: Playing game at server and client respectively

Expected behavior:

- All communication outside of playing a game, in particular the handshake and feature negotiation, works on both client and server in conjunction with the reference server and client, respectively.
- The server receives HELLO, LOGIN, QUEUE, LIST, MOVE, correctly
- The client receives HELLO, LOGIN, ALREADYLOGGEDIN, QUEUE, LIST, NEWGAME, MOVE, GAMEOVER, correctly

Testing result

1. Both the server and two clients start.
2. The client connects to the socket.



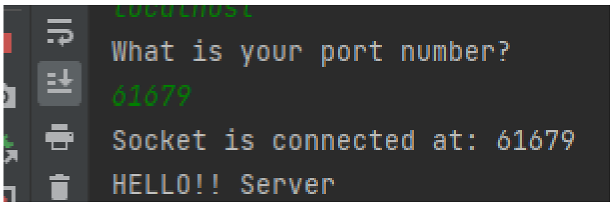
```
Run:  OthelloServerApplication
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2
Port number?
0
Start server at: 61679
Get connection to this socket: 127.0.0.1
Client Handler created and size: 1
```

```
Othello
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
What is your address?
localhost
What is your port number?
61679
Socket is connected at: 61679
```

```
Othello
C:\Users\syun1\.jdks\temurin-11.0.17\bi
What is your address?
localhost
What is your port number?
61679
Socket is connected at: 61679
```

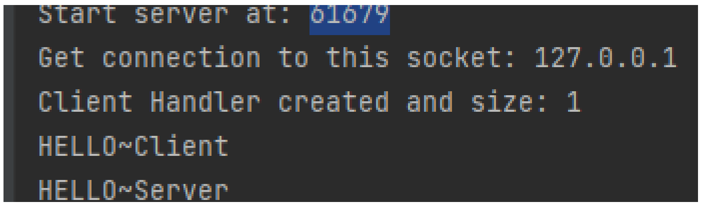
3. The client and the server send and receive the HELLO command.

The client sends.



```
localhost
What is your port number?
61679
Socket is connected at: 61679
HELLO!! Server
```

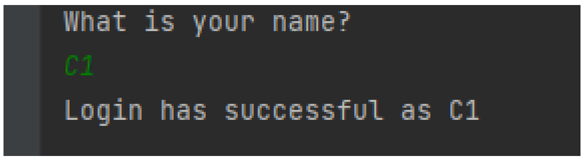
The server receives and sends.



```
Start server at: 61679
Get connection to this socket: 127.0.0.1
Client Handler created and size: 1
HELLO~Client
HELLO~Server
```

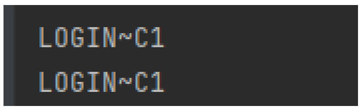
4. The client and the server send and receive the LOGIN command.

The client sends.



```
What is your name?
C1
Login has successful as C1
```

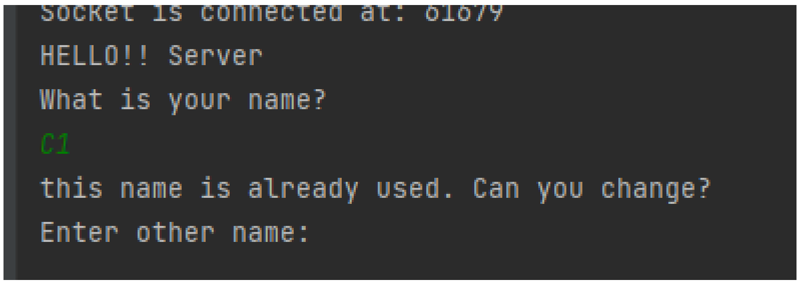
The server receives and sends.



```
LOGIN~C1
LOGIN~C1
```

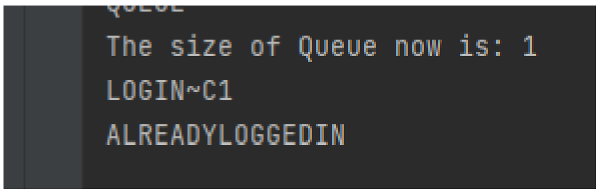
5. If the username is used, the server sends and the client receives the ALLREADYLOGGEDIN command.

The client tried to use the same name.



```
Socket is connected at: 61679
HELLO!! Server
What is your name?
C1
this name is already used. Can you change?
Enter other name:
```

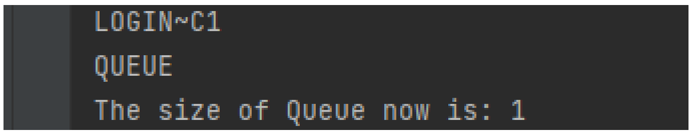
The server sends and the client receives.



```
QUEUE
The size of Queue now is: 1
LOGIN~C1
ALREADYLOGGEDIN
```

6. The client and the server send and receive the QUEUE command.

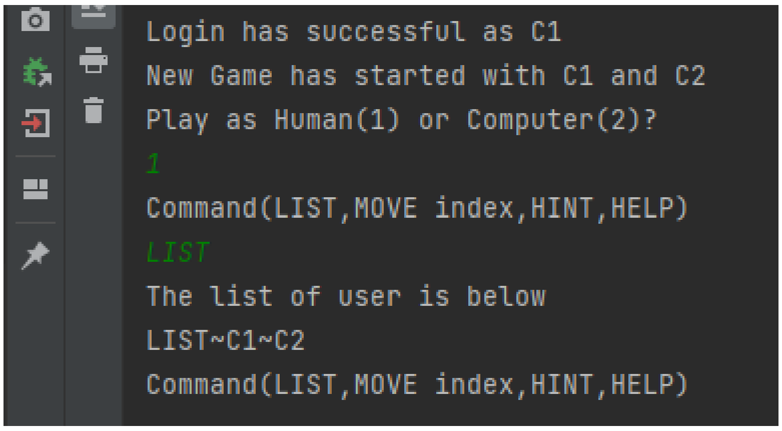
The client sends and the server receives.



```
LOGIN~C1
QUEUE
The size of Queue now is: 1
```

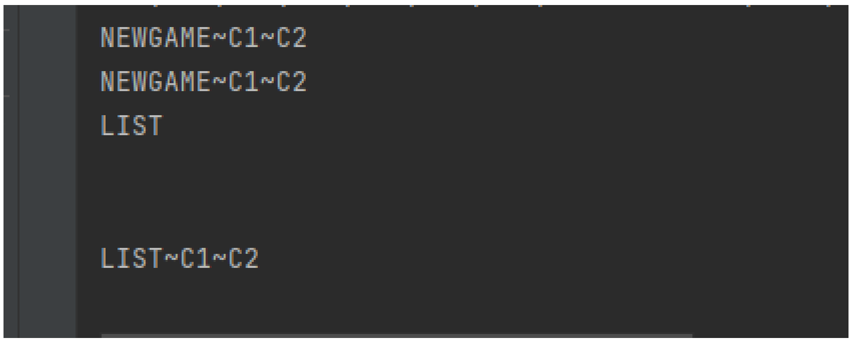
7. The client and the server send and receive the LIST command.

The clients send and receive.



```
Login has successful as C1
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
1
Command(LIST,MOVE index,HINT,HELP)
LIST
The list of user is below
LIST~C1~C2
Command(LIST,MOVE index,HINT,HELP)
```

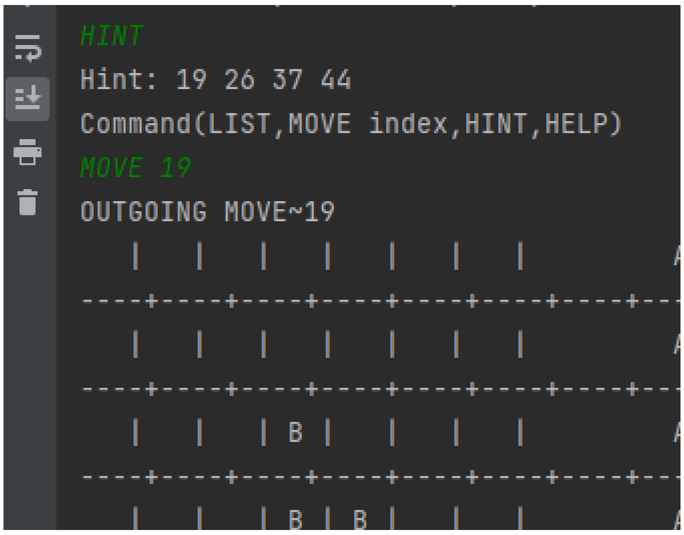
The server receives and sends.



```
NEWGAME~C1~C2
NEWGAME~C1~C2
LIST


LIST~C1~C2
```

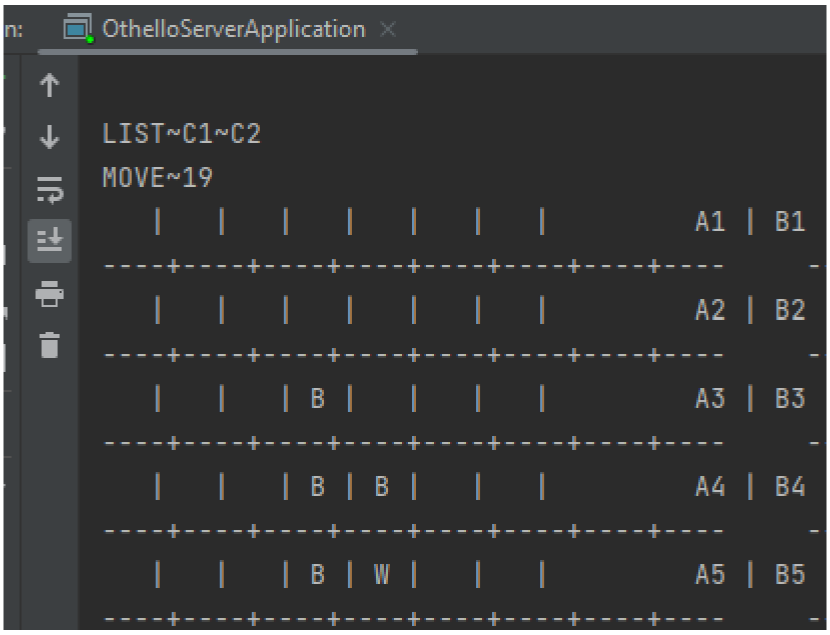
8. The client and the server send and receive the MOVE command.

The client sends the move.



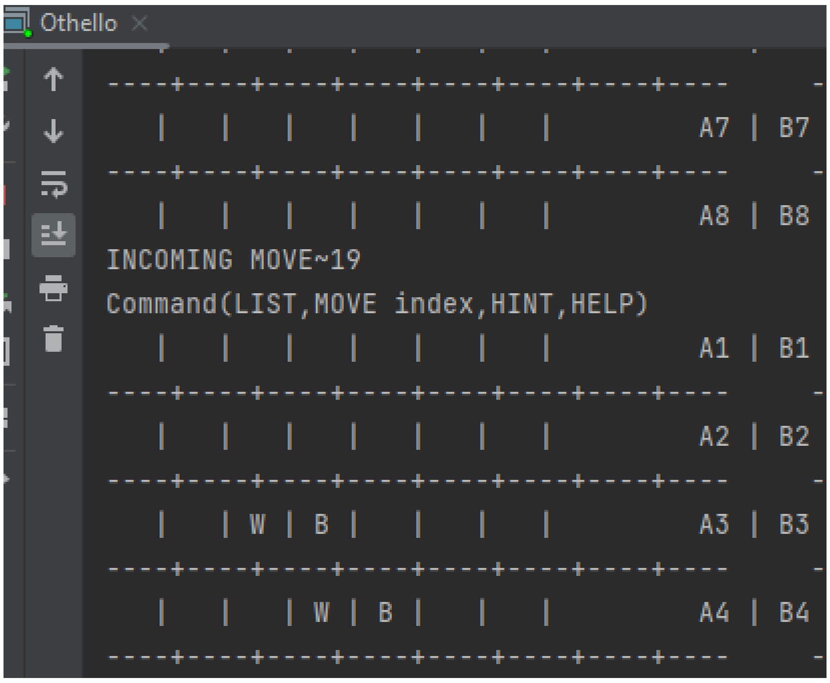
```
HINT
Hint: 19 26 37 44
Command(LIST,MOVE index,HINT,HELP)
MOVE 19
OUTGOING MOVE~19
    |    |    |    |    |    |    |          A
----+----+----+----+----+----+----+---
    |    |    |    |    |    |    |          A
----+----+----+----+----+----+----+---
    |    |    | B  |    |    |    |          A
----+----+----+----+----+----+----+---
    |    |    | B  | B  |    |    |          A
```

The server receives the move.



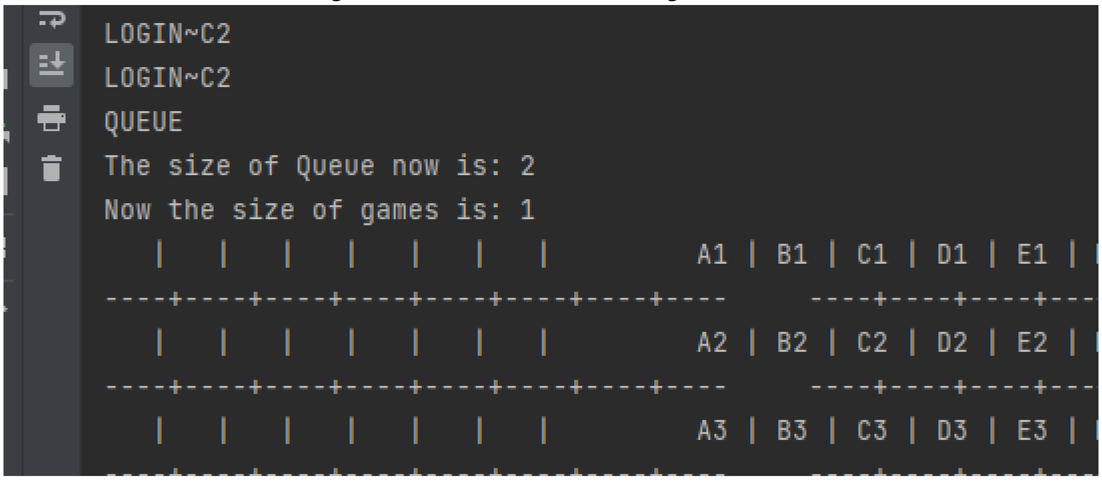
```
n:  OthelloServerApplication

LIST~C1~C2
MOVE~19
    |    |    |    |    |    |    |          A1 | B1
----+----+----+----+----+----+----+----          -
    |    |    |    |    |    |    |          A2 | B2
----+----+----+----+----+----+----+----          -
    |    |    | B  |    |    |    |          A3 | B3
----+----+----+----+----+----+----+----          -
    |    |    | B  | B  |    |    |          A4 | B4
----+----+----+----+----+----+----+----          -
    |    |    | B  | W  |    |    |          A5 | B5
----+----+----+----+----+----+----+----          -
```

The client receives the result of the move.



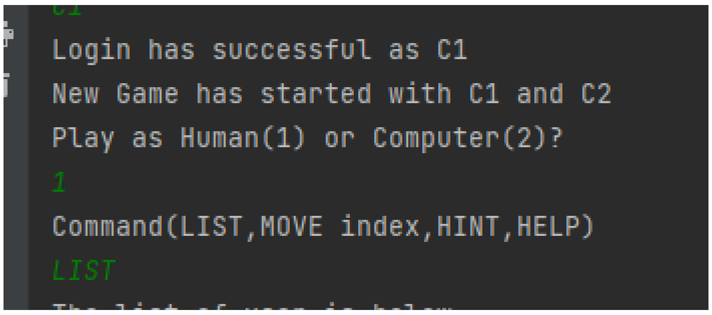
```
Othello
----+----+----+----+----+----+----+----     -
    |    |    |    |    |    |    |         A7 | B7
----+----+----+----+----+----+----+----     -
    |    |    |    |    |    |    |         A8 | B8
INCOMING MOVE~19
Command(LIST,MOVE index,HINT,HELP)
    |    |    |    |    |    |    |         A1 | B1
----+----+----+----+----+----+----+----     -
    |    |    |    |    |    |    |         A2 | B2
----+----+----+----+----+----+----+----     -
    |    | W  | B  |    |    |    |         A3 | B3
----+----+----+----+----+----+----+----     -
    |    |    | W  | B  |    |    |         A4 | B4
----+----+----+----+----+----+----+----     -
```

9. The server sends and the client receives the NEWGAME command.
The server makes a new game and sends a message.



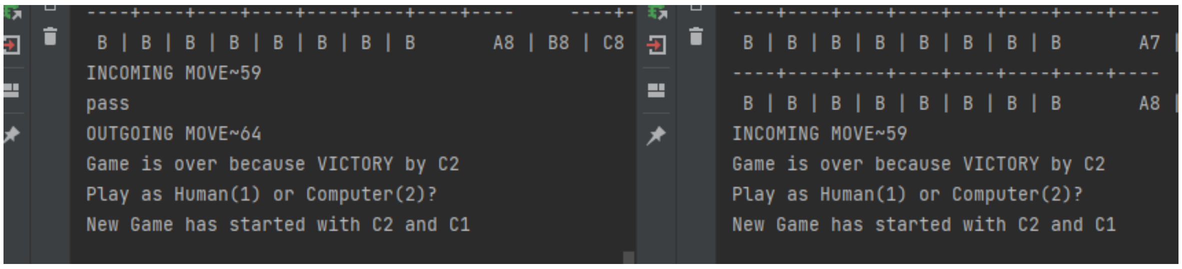
```
LOGIN~C2
LOGIN~C2
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
    |    |    |    |    |    |    |         A1 | B1 | C1 | D1 | E1 |
----+----+----+----+----+----+----+----     ----+----+----+---
    |    |    |    |    |    |    |         A2 | B2 | C2 | D2 | E2 |
----+----+----+----+----+----+----+----     ----+----+----+---
    |    |    |    |    |    |    |         A3 | B3 | C3 | D3 | E3 |
```

The client receives.



```
Login has successful as C1
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
1
Command(LIST,MOVE index,HINT,HELP)
LIST
```

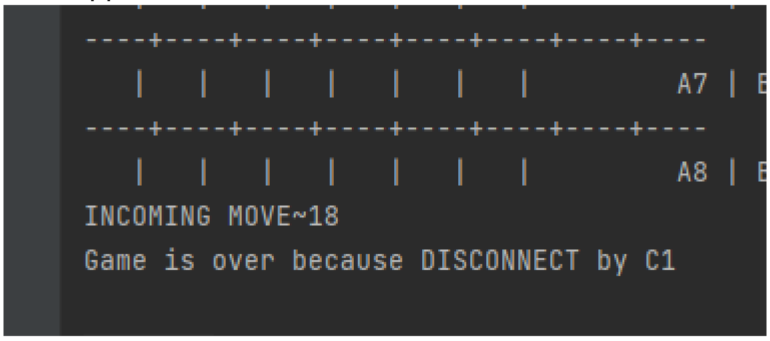
10. The server sends and the client receives the GAMEOVER command.
If the game has been finished by one of the clients won, the server sends and clients receive.



```
----+----+----+----+----+----+----+----          ----+-
  B | B | B | B | B | B | B | B        A8 | B8 | C8
INCOMING MOVE~59
pass
OUTGOING MOVE~64
Game is over because VICTORY by C2
Play as Human(1) or Computer(2)?
New Game has started with C2 and C1
```

```
----+----+----+----+----+----+----+----
  B | B | B | B | B | B | B | B        A7 |
----+----+----+----+----+----+----+----
  B | B | B | B | B | B | B | B        A8 |
INCOMING MOVE~59
Game is over because VICTORY by C2
Play as Human(1) or Computer(2)?
New Game has started with C2 and C1
```

If the opponent is disconnected, the server sends and the other client receives.



```
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A7 | B
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |         A8 | B
INCOMING MOVE~18
Game is over because DISCONNECT by C1
```

Testing Report for FR11

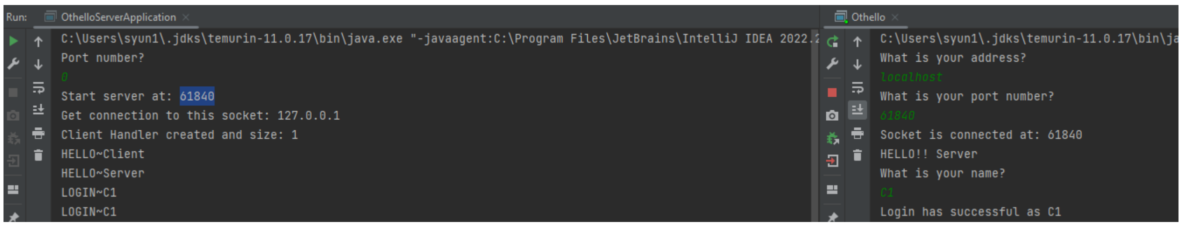
FR11: Client will finish without connection

Expected behavior:

- Whenever a client loses connection to a server, the client should gracefully terminate.

Testing result

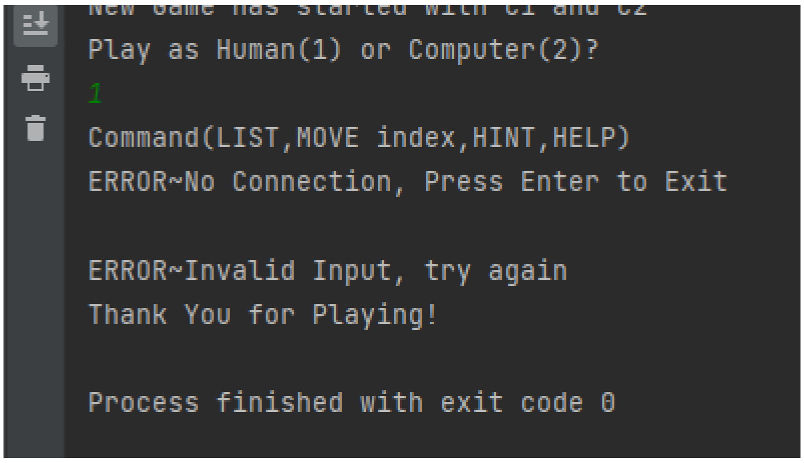
1. Make a connection between the server and the client.



```
Run:  OthelloServerApplication
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.2
Port number?
0
Start server at: 61840
Get connection to this socket: 127.0.0.1
Client Handler created and size: 1
HELLO~Client
HELLO~Server
LOGIN~C1
LOGIN~C1
```

```
Othello
C:\Users\syun1\.jdks\temurin-11.0.17\bin\ja
What is your address?
localhost
What is your port number?
61840
Socket is connected at: 61840
HELLO!! Server
What is your name?
C1
Login has successful as C1
```

2. The server disconnects the network.



```
New Game has started with C1 and C2
Play as Human(1) or Computer(2)?
1
Command(LIST,MOVE index,HINT,HELP)
ERROR~No Connection, Press Enter to Exit

ERROR~Invalid Input, try again
Thank You for Playing!

Process finished with exit code 0
```

Testing Report for FR12

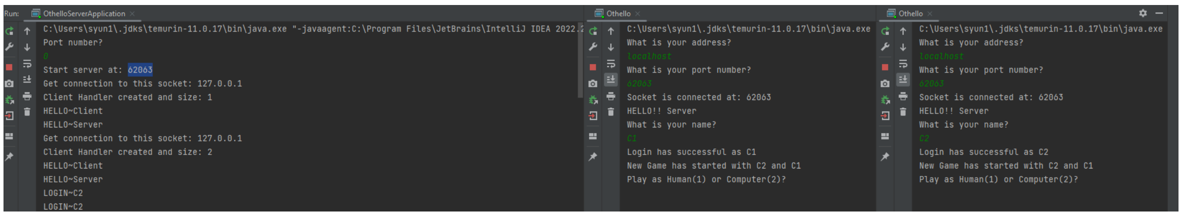
FR12: Server ends games and notify

Expected behavior:

- Whenever Client Disconnects During Game,the server should inform the other client(s) and end the game, allowing the other player to start a new game.

Testing result

1. Both the server and two clients start

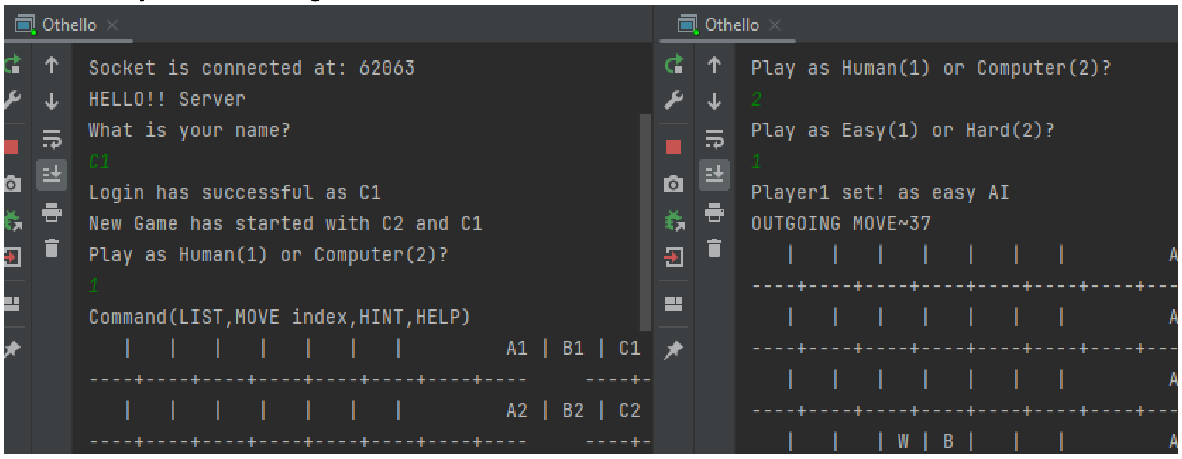


```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.
Port number?
0
Start server at: 62063
Get connection to this socket: 127.0.0.1
Client Handler created and size: 1
HELLO~Client
HELLO~Server
Get connection to this socket: 127.0.0.1
Client Handler created and size: 2
HELLO~Client
HELLO~Server
LOGIN~C2
LOGIN~C2
```

```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
What is your address?
localhost
What is your port number?
62063
Socket is connected at: 62063
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C2 and C1
Play as Human(1) or Computer(2)?
```

```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
What is your address?
localhost
What is your port number?
62063
Socket is connected at: 62063
HELLO!! Server
What is your name?
C2
Login has successful as C2
New Game has started with C2 and C1
Play as Human(1) or Computer(2)?
```

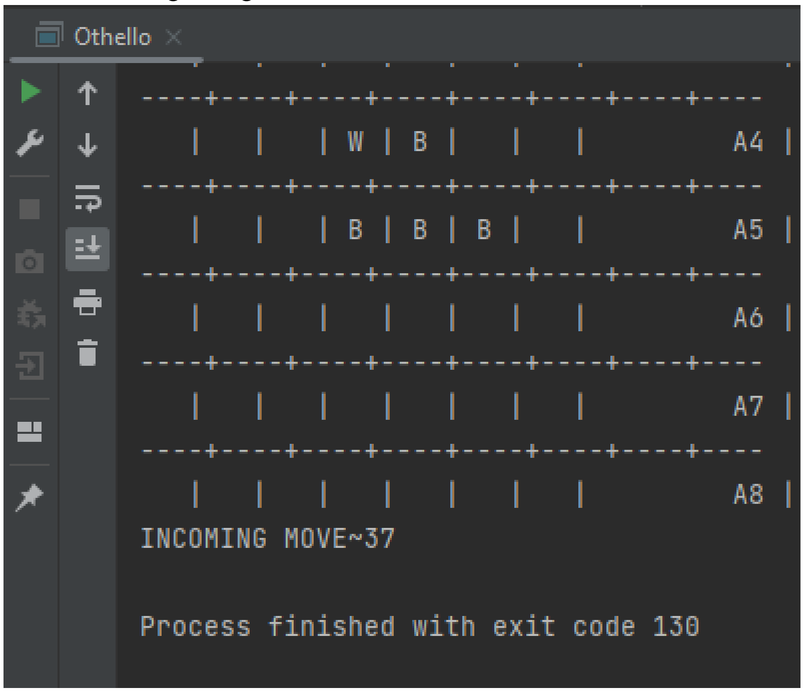
2. Play a standard game.



```
Socket is connected at: 62063
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C2 and C1
Play as Human(1) or Computer(2)?
1
Command(LIST,MOVE index,HINT,HELP)
    |    |    |    |    |    |    |        A1  | B1  | C1
----+----+----+----+----+----+----+----     ----+-
    |    |    |    |    |    |    |        A2  | B2  | C2
----+----+----+----+----+----+----+----     ----+-
```

```
Play as Human(1) or Computer(2)?
2
Play as Easy(1) or Hard(2)?
1
Player1 set! as easy AI
OUTGOING MOVE~37
    |    |    |    |    |    |    |        A
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |        A
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |        A
----+----+----+----+----+----+----+----
    |    |    | W  | B  |    |    |        A
```

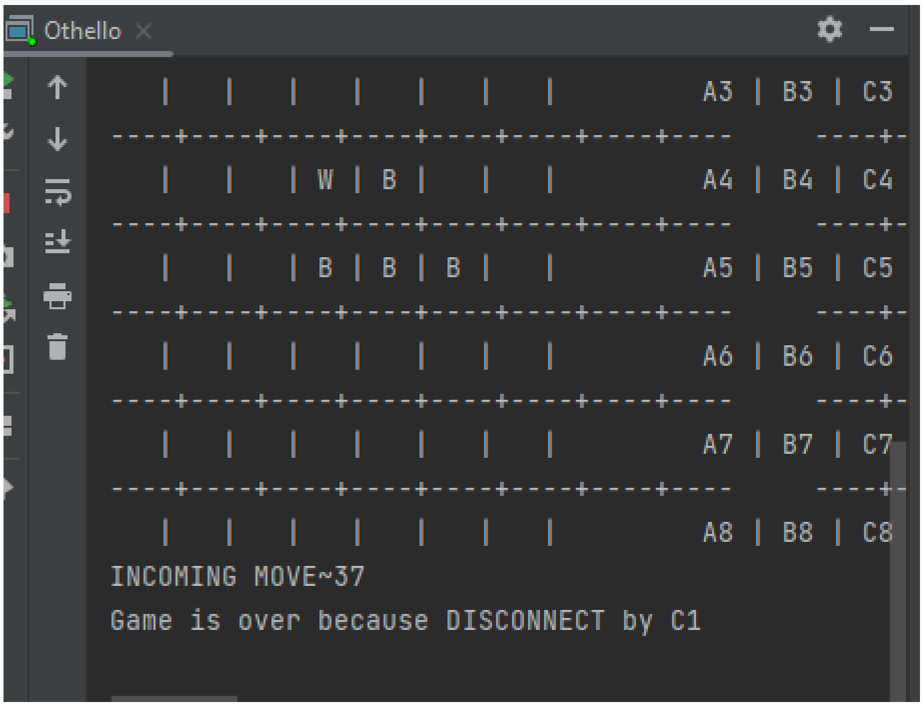
3. During the game one of the clients disconnects the connection with the network.



```
----+----+----+----+----+----+----+----
    |    |    | W  | B  |    |    |        A4  |
----+----+----+----+----+----+----+----
    |    |    | B  | B  | B  |    |        A5  |
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |        A6  |
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |        A7  |
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |        A8  |
INCOMING MOVE~37

Process finished with exit code 130
```

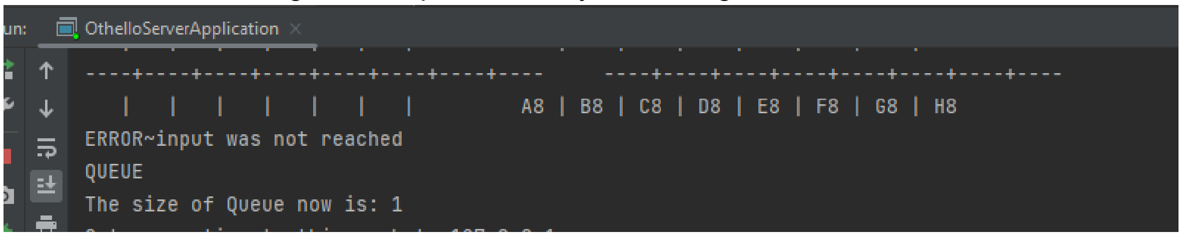
4. The other client can start a new game after the opponent comes to the queue.

The rest client receives messages from the server



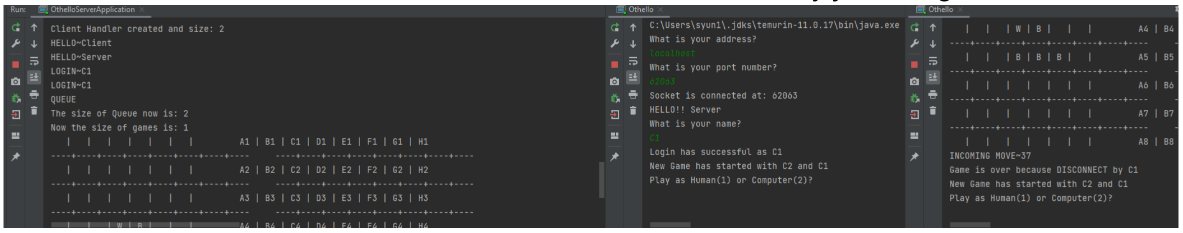
```
    |    |    |    |    |    |    |          A3 | B3 | C3
----+----+----+----+----+----+----+----      ----+-
    |    |    | W  | B  |    |    |          A4 | B4 | C4
----+----+----+----+----+----+----+----      ----+-
    |    |    | B  | B  | B  |    |          A5 | B5 | C5
----+----+----+----+----+----+----+----      ----+-
    |    |    |    |    |    |    |          A6 | B6 | C6
----+----+----+----+----+----+----+----      ----+-
    |    |    |    |    |    |    |          A7 | B7 | C7
----+----+----+----+----+----+----+----      ----+-
    |    |    |    |    |    |    |          A8 | B8 | C8
INCOMING MOVE~37
Game is over because DISCONNECT by C1
```

The rest of the clients go to the queue directly after the game is over.



```
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |          A8 | B8 | C8 | D8 | E8 | F8 | G8 | H8
ERROR~input was not reached
QUEUE
The size of Queue now is: 1
```

After a new client comes, the rest of the clients can immediately join the game.



```
Client Handler created and size: 2
HELLO~Client
HELLO~Server
LOGIN~C1
LOGIN~C1
QUEUE
The size of Queue now is: 2
Now the size of games is: 1
    |    |    |    |    |    |    |          A1 | B1 | C1 | D1 | E1 | F1 | G1 | H1
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |          A2 | B2 | C2 | D2 | E2 | F2 | G2 | H2
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |          A3 | B3 | C3 | D3 | E3 | F3 | G3 | H3
----+----+----+----+----+----+----+----      ----+----+----+----+----+----+----+----
```

```
C:\Users\syun1\.jdks\temurin-11.0.17\bin\java.exe
What is your address?
localhost
What is your port number?
62063
Socket is connected at: 62063
HELLO!! Server
What is your name?
C1
Login has successful as C1
New Game has started with C2 and C1
Play as Human(1) or Computer(2)?
```

```
    |    |    | W  | B  |    |    |          A4 | B4
----+----+----+----+----+----+----+----
    |    |    | B  | B  | B  |    |          A5 | B5
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |          A6 | B6
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |          A7 | B7
----+----+----+----+----+----+----+----
    |    |    |    |    |    |    |          A8 | B8
INCOMING MOVE~37
Game is over because DISCONNECT by C1
New Game has started with C2 and C1
Play as Human(1) or Computer(2)?
```

Test Strategy:

a) Scope of testing:

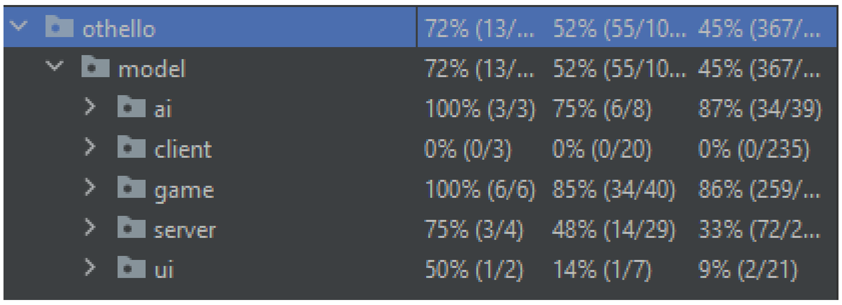
- Simple Othello game logic program that is able to act as a virtual board and game master at the same time, playing the game with inputs from players.
- An “AI” player, acting as a player, with two difficulty settings.
- An easy to use and clear game client that receives all the sent information and shows the user all the necessary visuals to play the game.
- A server that connects two clients to a game. The game is executed on the server itself and is played with the input from the connected clients.
- Fluid server-client infrastructure, with insurance that there will be close to none data loss or crashes between the two.

b) Risks
   - An individual could breach the server and either cheat by altering the board in illegal ways or simply break the game.
   - A lot of traffic could overload the connection/server and cause lag.
   - Inefficient code could slow down the execution of the game making the experience slow.
   - Badly implemented "AI" player could make facing it either too easy or too hard.
   - Server code that does not respect thread safety could lead to data being altered in unintended ways making the game unplayable.

c) Test levels
   - Singular player: the perspective of the game for a regular player joining from their machine for a game.
   - Server: everything that takes place directly on the server, meaning any "backend" processes happening exclusively on it.
   - Game: perspective of two clients/players and their thread on the server all taking part in one game instance.
   - Whole system: all the players/clients in their games, all taking place on the server.

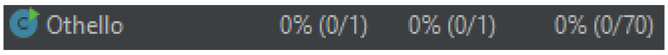
d) Test types
   - Functional: tests that make sure everything works in the technical sense, meaning there are no bugs, glitches or unintended side effects that would affect the service negatively
   - Security: test if all the exchanges are properly encrypted and if confidential data cannot be reached from the outside by unauthorized parties.
   - Performance: see if the game runs smoothly, without too many stutters or lag while also putting as little strain on the users machine as possible.
   - User Interaction: test with users, see if the UI feels intuitive to navigate, if the learning curve of using it isn't too hard and the process isn't annoying.

e) Metrics
   - Coverage -

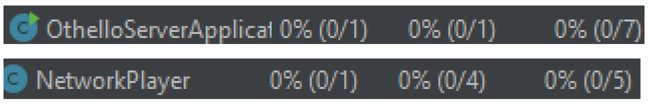


othello	72% (13/...	52% (55/10...	45% (367/...
model	72% (13/...	52% (55/10...	45% (367/...
ai	100% (3/3)	75% (6/8)	87% (34/39)
client	0% (0/3)	0% (0/20)	0% (0/235)
game	100% (6/6)	85% (34/40)	86% (259/...
server	75% (3/4)	48% (14/29)	33% (72/2...
ui	50% (1/2)	14% (1/7)	9% (2/21)

   - Our tests have scored a 72% of class coverage and 52% for the whole project. While we have not reached the target of 90% there are a few explanations/
   - Firstly as will be explained later some classes were completely ignored in the tests simply because we didn't deem it necessary, such as :

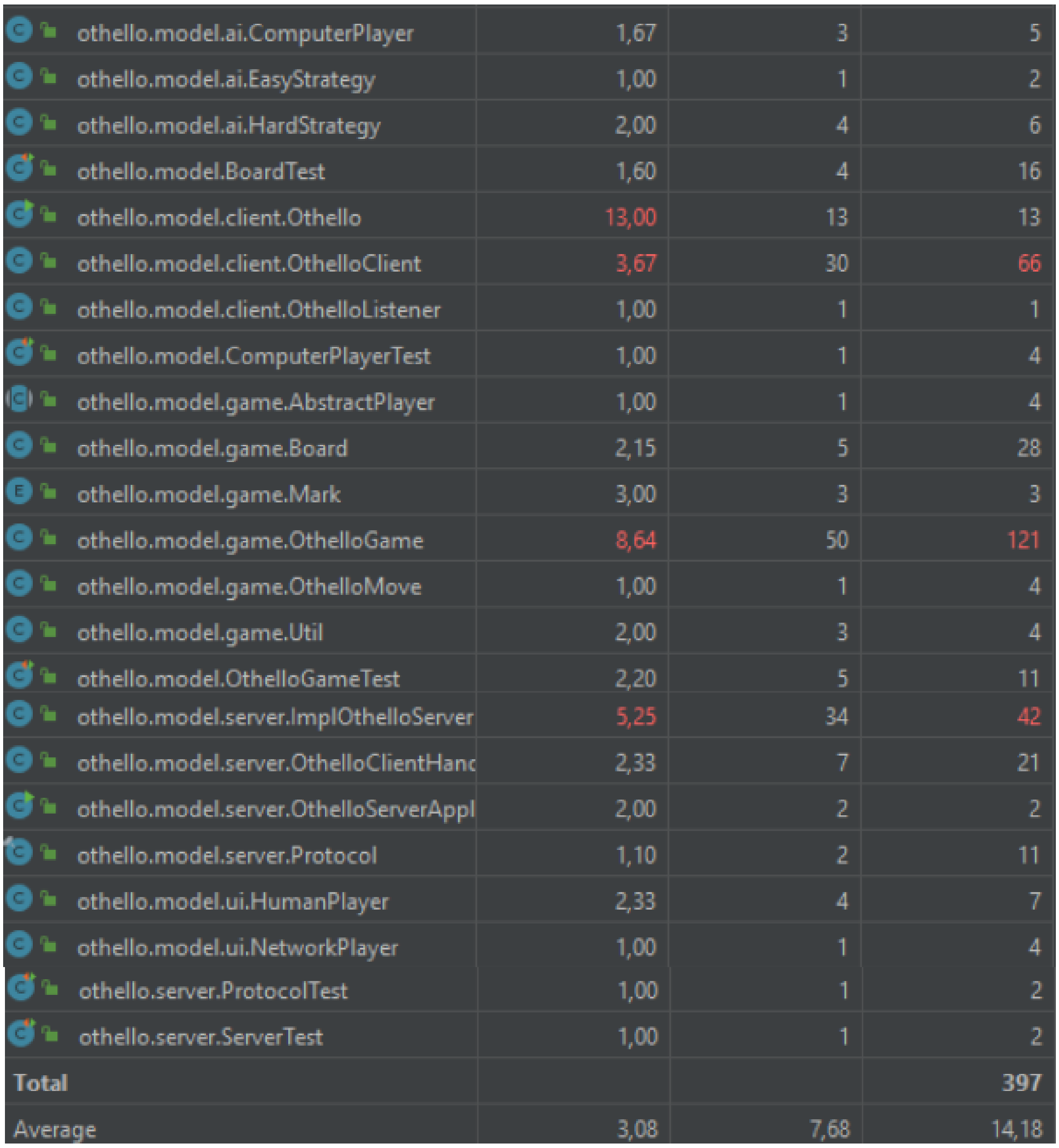


Othello	0% (0/1)	0% (0/1)	0% (0/70)



OthelloServerApplicat	0% (0/1)	0% (0/1)	0% (0/7)
NetworkPlayer	0% (0/1)	0% (0/4)	0% (0/5)

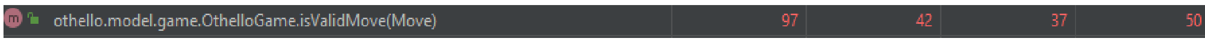
- Another factor was our lack of time, another lesson for the future. The biggest regret was that we have never managed to put together a proper Client Test. One reason was that we found it hard to test client behavior since all of its output is being printed via System.out and we didn't know how to read that. Another factor was a simple lack of time to figure out how to make a test that would measure clients functionality well.
- Cyclomatic Complexity:



othello.model.ai.ComputerPlayer	1,67	3	5
othello.model.ai.EasyStrategy	1,00	1	2
othello.model.ai.HardStrategy	2,00	4	6
othello.model.BoardTest	1,60	4	16
othello.model.client.Othello	13,00	13	13
othello.model.client.OthelloClient	3,67	30	66
othello.model.client.OthelloListener	1,00	1	1
othello.model.ComputerPlayerTest	1,00	1	4
othello.model.game.AbstractPlayer	1,00	1	4
othello.model.game.Board	2,15	5	28
othello.model.game.Mark	3,00	3	3
othello.model.game.OthelloGame	8,64	50	121
othello.model.game.OthelloMove	1,00	1	4
othello.model.game.Util	2,00	3	4
othello.model.OthelloGameTest	2,20	5	11
othello.model.server.ImplOthelloServer	5,25	34	42
othello.model.server.OthelloClientHanc	2,33	7	21
othello.model.server.OthelloServerAppl	2,00	2	2
othello.model.server.Protocol	1,10	2	11
othello.model.ui.HumanPlayer	2,33	4	7
othello.model.ui.NetworkPlayer	1,00	1	4
othello.server.ProtocolTest	1,00	1	2
othello.server.ServerTest	1,00	1	2
Total			**397**
Average	3,08	7,68	14,18

  - Most of the complexities across the board are decent, averaging a OC of 2.
  - Unfortunately there are a few outliers with the Othello class reaching 13. This is likely caused by the multiple loops that we used to run the application for different cases.

- The next biggest outlier is the OthelloGame class. We believe this is caused by the over-crowded method that checks for valid moves, which scored a 97. One look at the structure of the methods code explains this very well, multiple nested conditional statements each with multiple conditions and long processes inside of them, all stacked in one method increased the complexity drastically. This is a lesson for the future, to keep the design more modular and separate those cases into different methods/classes.



```
othello.model.game.OthelloGame.isValidMove(Move)     97     42     37     50
```

- Lastly there are the Client and Server classes, which logically it makes sense that they are slightly more involved than the rest since they are the main logic operating units of the client/server architecture of our game.
- Overall, our project scored a 3,39 average complexity which we find to be sufficient and while there definitely could be improvements, the performance is decent and stable enough.

f) Least tested components
   - “client” package of classes is the least tested.
   - apart from system tests and the protocol test, we didn’t test these functionality.
   - It may cause trouble in client function. Especially at the unit level, methods like the sendCommand method may not work correctly.

g)
   - Client UI - the class “Othello” is not involved in any of the tests since it is a very simple TUI running in a loop and it’s lack of complex methods or such we hardly thought it deserves thorough testing
   - Server UI - similarly the class “ServerApplication” influences the functionalities of the server very minimally, only serves to get the port number and print out messages, so we didn’t feel testing it was needed
   - Util - the “Util” class is a class that stores random methods used for calculating specific things. The functionality it’s methods provide are very basic and don’t need interaction with any other component of the system so we didn’t feel the need to test it in other ways then making a temporary main and running the commands with few “test” inputs.

Test Plan

a) Schedule :
   - Game logic and tests for it should be completed before the Midway Submission deadline.
   - Testing of any feature should be completed within 2 days of its first implementation or any future adjustments of it.
   - The whole testing process and any final checks should be completed before the 30th of January.

b) Deliverables:
   - Before testing: test schedule and plan document

- During testing: data required for testing, testing tools
- After testing: test results, logs, bug reports, summary of the process

c) Objectives:

- The main objective of the testing is to ensure a service that is functional and secure, meaning there are slight to none chances of bugs and crashes as well as no possibility of data loss.
- The other objective is a product that is easy and satisfying to use, which doesn’t strain the user to find any function/data they might want to access and makes the navigation process intuitive and as a result satisfying as a whole.
- The testing units should have an average of 90% coverage of the project.

Unit Tests

a) BoardTest

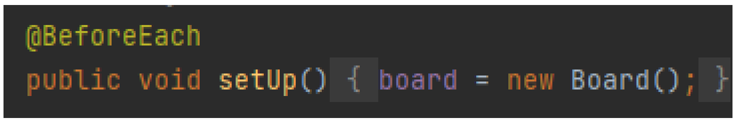
Testing Report for Board

The Board should imitate a real life Othello board near perfect

Expected behavior:

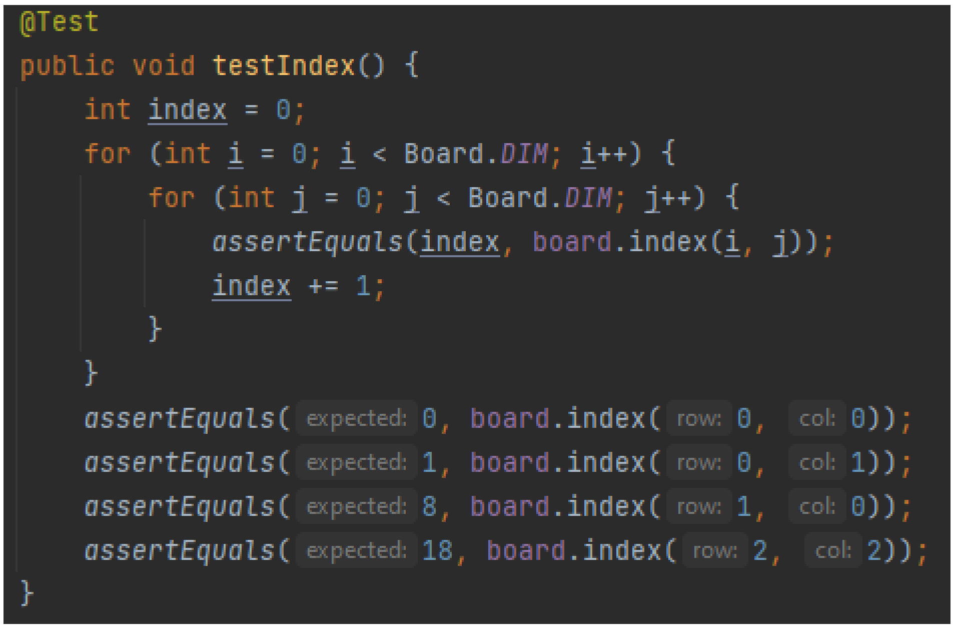
- All the tiles indexes go from right to left on an 8x8 board
- All the tiles are numbered from 0 to 63
- All the tiles can be set as either BLACK, WHITE or EMPTY
- The program can also access the mark value of each field via a getter
- The board can test each field to see whether its empty
- All of the previous properties can be accessed and tested via single integer index input as well as a row + column input
- The deepCopy function of the Board returns an identical copy of the given board

Actual behavior:



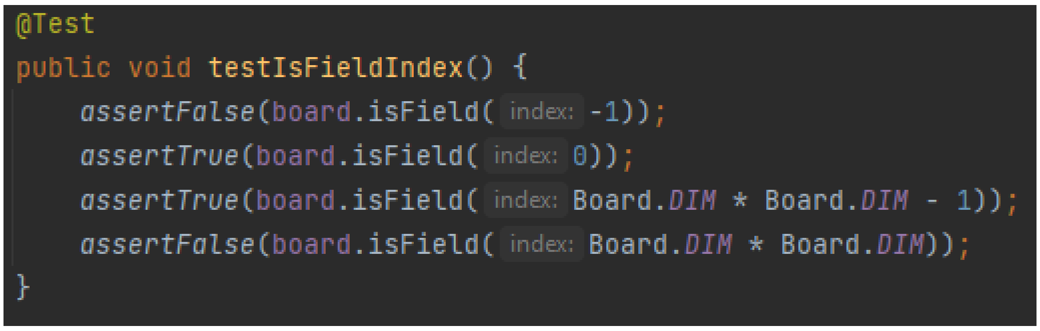
```java
@BeforeEach
public void setUp() { board = new Board(); }
```

- The setup creates a new Board object instance



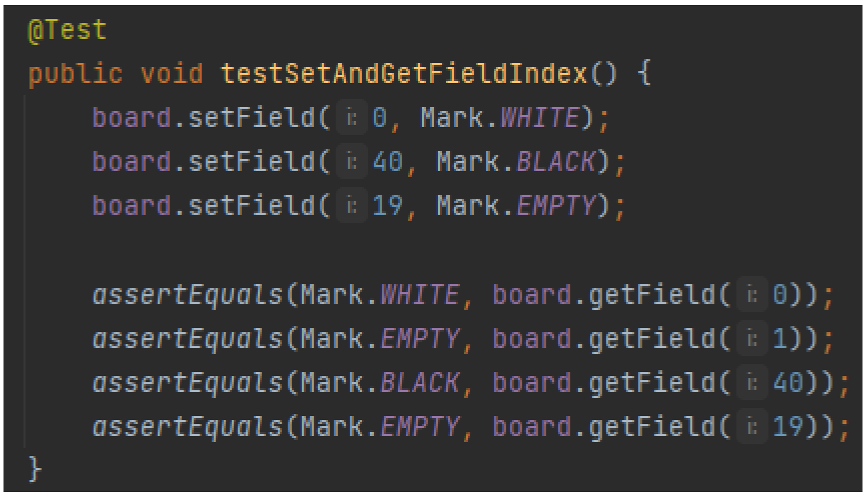
```java
@Test
public void testIndex() {
    int index = 0;
    for (int i = 0; i < Board.DIM; i++) {
        for (int j = 0; j < Board.DIM; j++) {
            assertEquals(index, board.index(i, j));
            index += 1;
        }
    }
    assertEquals( expected: 0, board.index( row: 0,  col: 0));
    assertEquals( expected: 1, board.index( row: 0,  col: 1));
    assertEquals( expected: 8, board.index( row: 1,  col: 0));
    assertEquals( expected: 18, board.index( row: 2,  col: 2));
}
```

- testIndex loops over 8 times for vertical and horizontal dimensions
- For each instance in the loop it checks if the row + column input for i and j checks out with the actual indexing of the board
- With this it's clear that for example the index of the second column and second row field is 18



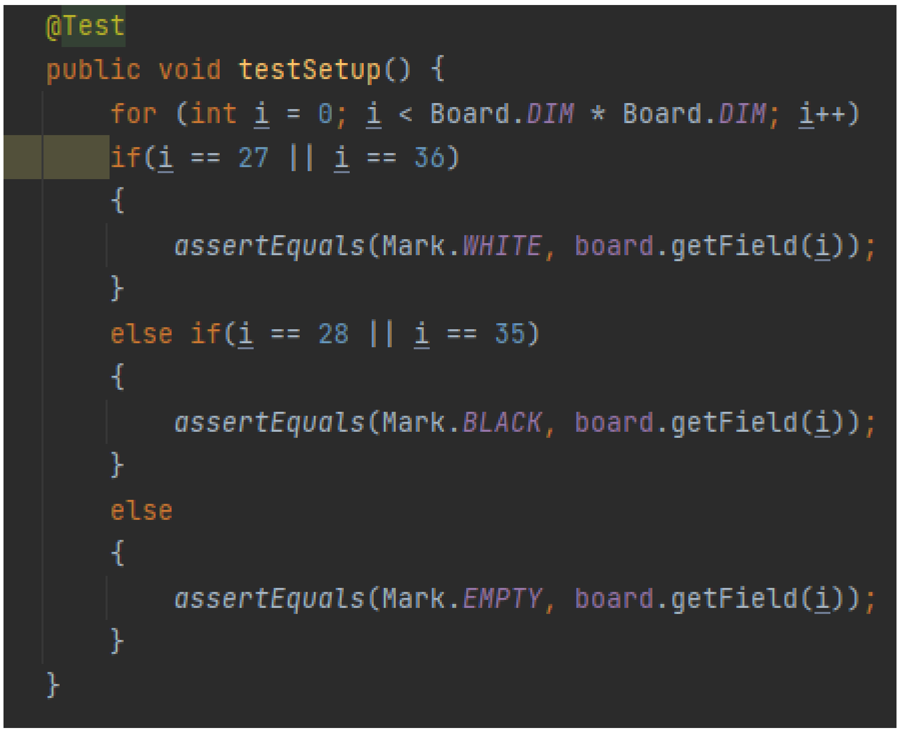
```java
@Test
public void testIsFieldIndex() {
    assertFalse(board.isField( index: -1));
    assertTrue(board.isField( index: 0));
    assertTrue(board.isField( index: Board.DIM * Board.DIM - 1));
    assertFalse(board.isField( index: Board.DIM * Board.DIM));
}
```

- When using the index –1 with isField it returns false
- Index 0 on the other hand returns true
- Same with 63, which is the highest possible index number
- Since indexing starts at 0, inputting 64 into isField returns false



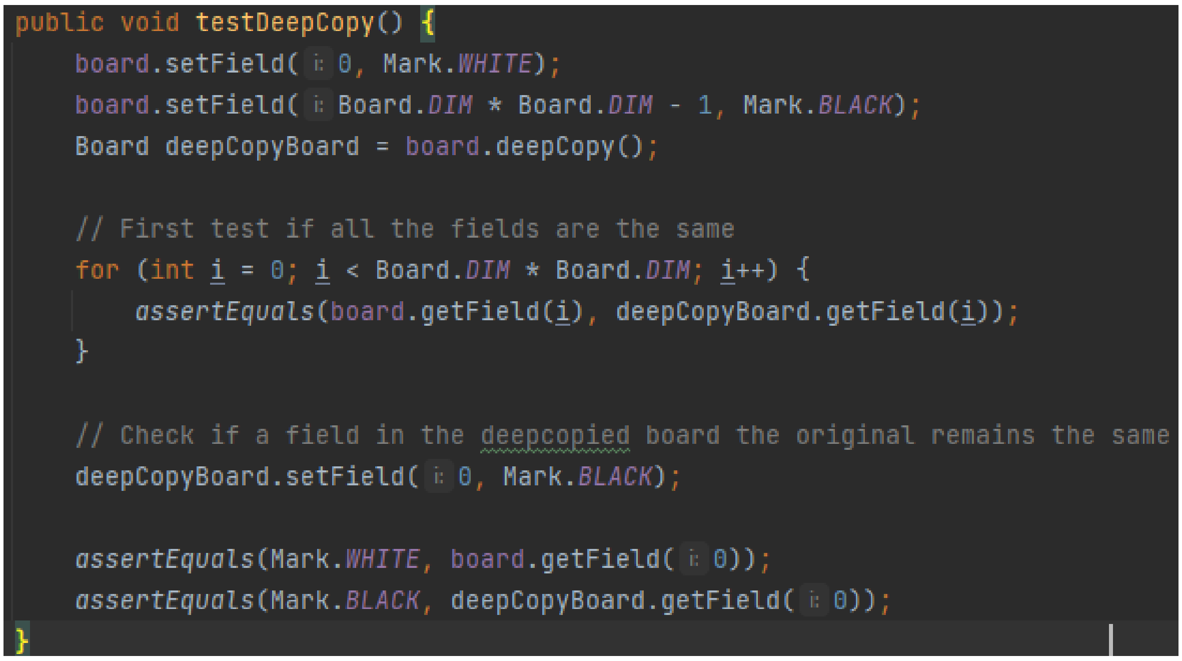
```java
@Test
public void testSetAndGetFieldIndex() {
    board.setField( i: 0, Mark.WHITE);
    board.setField( i: 40, Mark.BLACK);
    board.setField( i: 19, Mark.EMPTY);

    assertEquals(Mark.WHITE, board.getField( i: 0));
    assertEquals(Mark.EMPTY, board.getField( i: 1));
    assertEquals(Mark.BLACK, board.getField( i: 40));
    assertEquals(Mark.EMPTY, board.getField( i: 19));
}
```

- Since setField set index 0 on the board to white, getField(0) returns white as well
- Since setField set index 40 on the board to black, getField(40) returns black as well
- Since setField set index 19 on the board to empty, getField(19) returns empty as well
- Since field index 1 hasn't been set to anything, getField(1) returns empty



```java
@Test
public void testSetup() {
    for (int i = 0; i < Board.DIM * Board.DIM; i++)
    if(i == 27 || i == 36)
    {
        assertEquals(Mark.WHITE, board.getField(i));
    }
    else if(i == 28 || i == 35)
    {
        assertEquals(Mark.BLACK, board.getField(i));
    }
    else
    {
        assertEquals(Mark.EMPTY, board.getField(i));
    }
}
```

- Since board construction works well, all the fields except the select four of a brand new board instance are empty
- Pieces 27 and 36 are white
- 28 and 35 are black

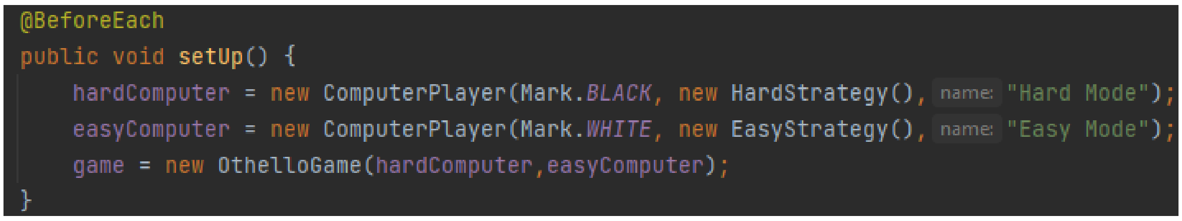


```java
public void testDeepCopy() {
    board.setField( i: 0, Mark.WHITE);
    board.setField( i: Board.DIM * Board.DIM - 1, Mark.BLACK);
    Board deepCopyBoard = board.deepCopy();

    // First test if all the fields are the same
    for (int i = 0; i < Board.DIM * Board.DIM; i++) {
        assertEquals(board.getField(i), deepCopyBoard.getField(i));
    }

    // Check if a field in the deepcopied board the original remains the same
    deepCopyBoard.setField( i: 0, Mark.BLACK);

    assertEquals(Mark.WHITE, board.getField( i: 0));
    assertEquals(Mark.BLACK, deepCopyBoard.getField( i: 0));
}
```

- Since deepCopy should produce an exact copy, the test compares each field of both boards at the same index and checks if they are equal
- Then setField changes the mark in deepCopy at index 0 to black
- Then checks if the change has only been made to the copy

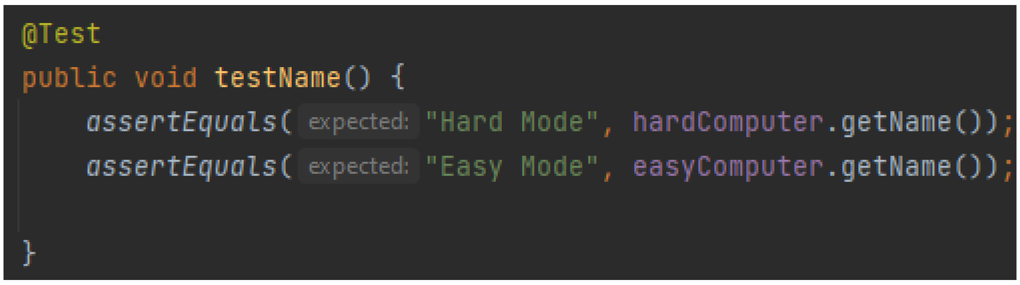
b) ComputerPlayerTest

Testing Report for Computer Player
The Computer Player should have all the attributes of a Player and two different difficulty levels. He should "produce" appropriate moves based on that level and send them to the game.
Expected behavior: - The Hard and Easy modes should have the difficulty in their name respectively - Each Computer Player should be able to store a mark which later can be reliably accessed - The moves produced by each Computer Player should always be valid
Actual behavior:



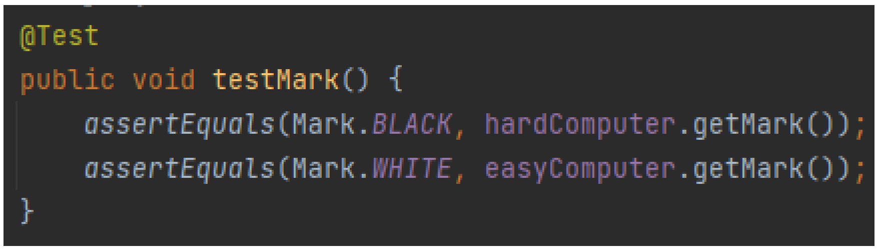
```java
@BeforeEach
public void setUp() {
    hardComputer = new ComputerPlayer(Mark.BLACK, new HardStrategy(), name: "Hard Mode");
    easyComputer = new ComputerPlayer(Mark.WHITE, new EasyStrategy(), name: "Easy Mode");
    game = new OthelloGame(hardComputer,easyComputer);
}
```

- The setup creates two Computer Players, one that implements the "Easy" strategy, the other one "Hard"
- Then it creates a new game with both of them as players



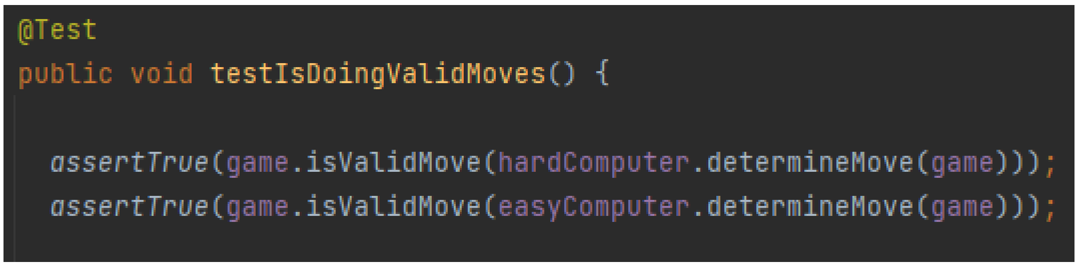
```java
@Test
public void testName() {
    assertEquals( expected: "Hard Mode", hardComputer.getName());
    assertEquals( expected: "Easy Mode", easyComputer.getName());

}
```

- Since both “modes” have predefined names they both return them when getName is used



```java
@Test
public void testMark() {
    assertEquals(Mark.BLACK, hardComputer.getMark());
    assertEquals(Mark.WHITE, easyComputer.getMark());
}
```

- Since when being created both players had a mark assigned, their getMark returns the same one



```java
@Test
public void testIsDoingValidMoves() {

  assertTrue(game.isValidMove(hardComputer.determineMove(game)));
  assertTrue(game.isValidMove(easyComputer.determineMove(game)));
```

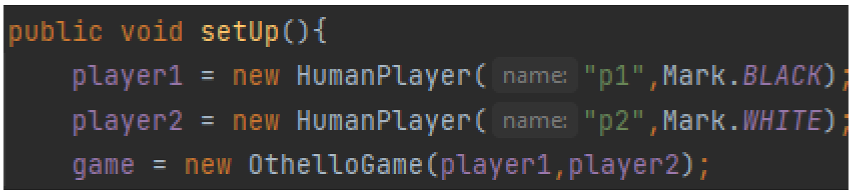
- Both are true since the moves returned by the corresponding implementation of the determineMove function which any Computer Player uses to make a move, are valid in the game

c) OthelloGameTest

Testing Report Othello Game
The Othello game should emulate all the logic and rules of the real board game without errors.
Expected behavior: - The game should be able to reliably detect that the game is over once there are no other valid moves left for either player - The game should be able to return the appropriate player based on whose turn is it, starting with player1 - The game should be able to without error recognize the winner of a game once it has finished - If the game has finished and both players have an equal number of tiles, the game should consider it a draw and leave the winner as null in consequence - The game should correctly produce a list of valid moves for the current game stage

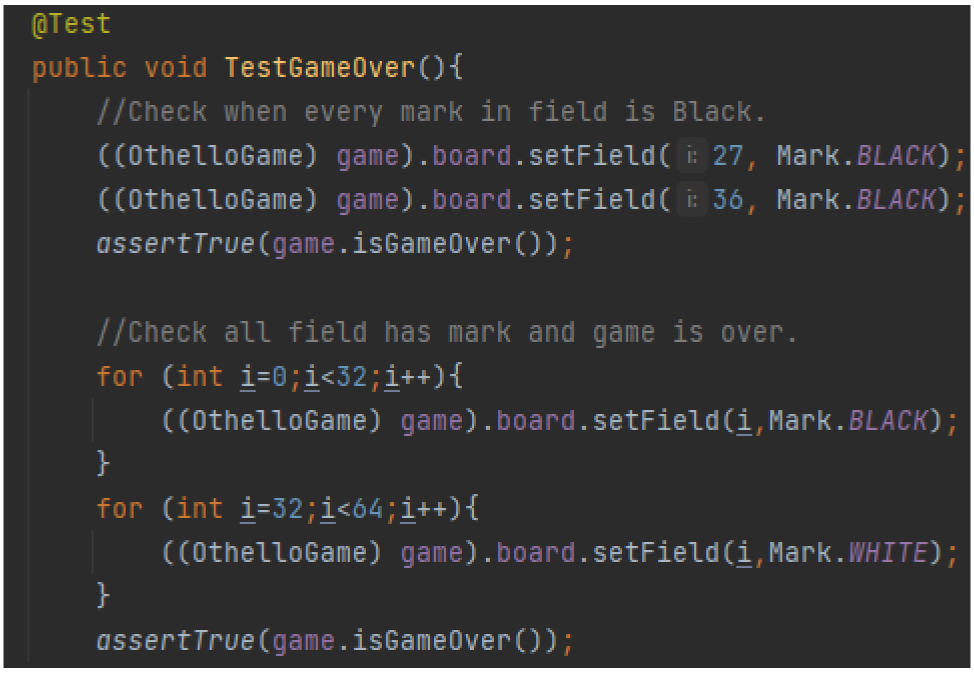
- All of the above should also be valid for Computer Players

Actual behavior:



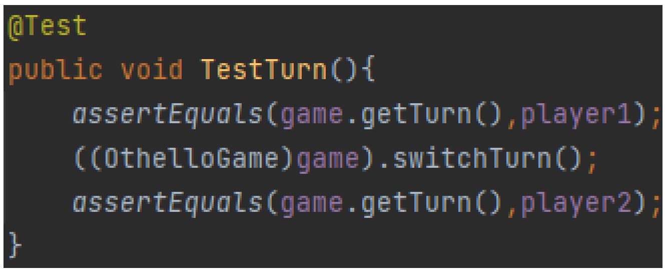
```java
public void setUp(){
    player1 = new HumanPlayer( name: "p1",Mark.BLACK);
    player2 = new HumanPlayer( name: "p2",Mark.WHITE);
    game = new OthelloGame(player1,player2);
```

- The setup creates two human players and then makes a game with them



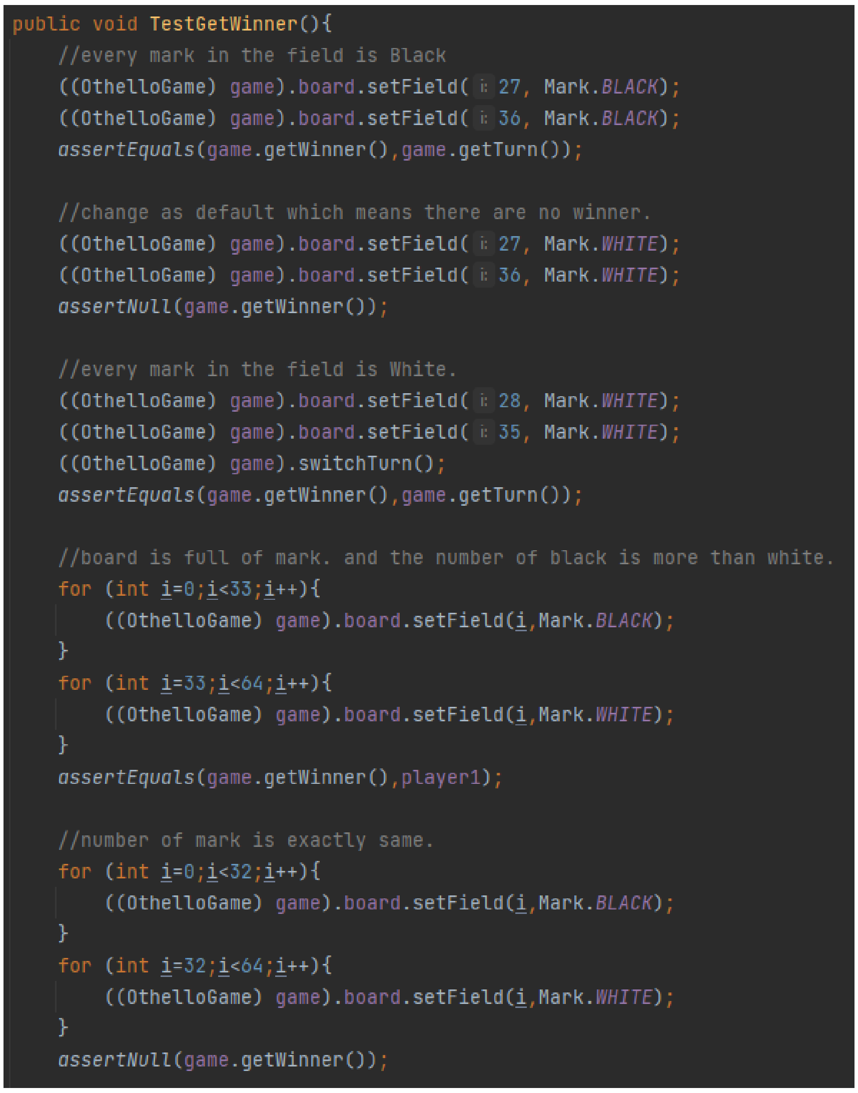
```java
 @Test
 public void TestGameOver(){
     //Check when every mark in field is Black.
     ((OthelloGame) game).board.setField( i: 27, Mark.BLACK);
     ((OthelloGame) game).board.setField( i: 36, Mark.BLACK);
     assertTrue(game.isGameOver());

     //Check all field has mark and game is over.
     for (int i=0;i<32;i++){
         ((OthelloGame) game).board.setField(i,Mark.BLACK);
     }
     for (int i=32;i<64;i++){
         ((OthelloGame) game).board.setField(i,Mark.WHITE);
     }
     assertTrue(game.isGameOver());
```

- This test first sets all the fields on the board to be black
- Then checks if the game detects that it's over since there are no possible moves for either player
- Next it fills up half of the board with black second half with white pieces
- Then checks if the game knows it's game over since the board is full



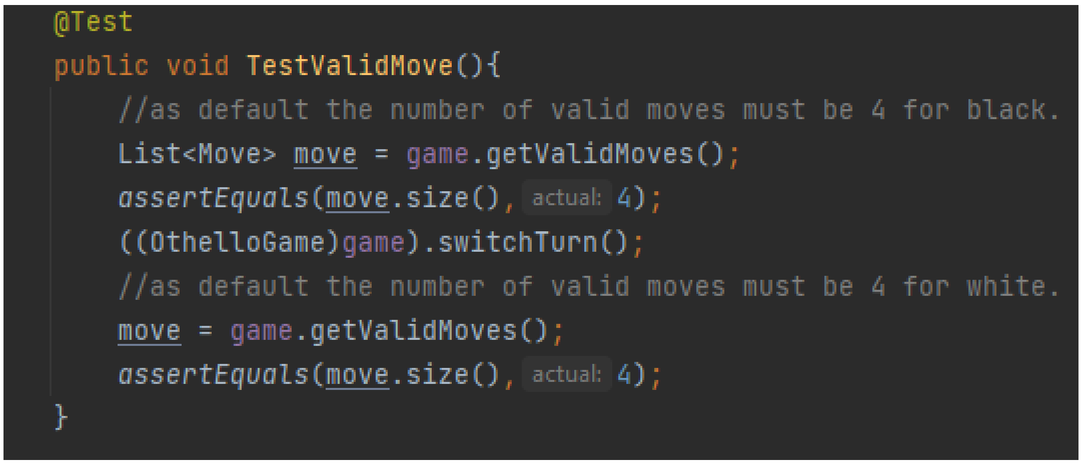
```java
@Test
public void TestTurn(){
    assertEquals(game.getTurn(),player1);
    ((OthelloGame)game).switchTurn();
    assertEquals(game.getTurn(),player2);
}
```

- This test first checks if the player stored as “player1” has the first turn of the game
- Then it uses the switchTurn command to change it to “player2”
- And then checks if the switchTurn in fact worked and if getTurn returns



```java
public void TestGetWinner(){
    //every mark in the field is Black
    ((OthelloGame) game).board.setField( i: 27, Mark.BLACK);
    ((OthelloGame) game).board.setField( i: 36, Mark.BLACK);
    assertEquals(game.getWinner(),game.getTurn());

    //change as default which means there are no winner.
    ((OthelloGame) game).board.setField( i: 27, Mark.WHITE);
    ((OthelloGame) game).board.setField( i: 36, Mark.WHITE);
    assertNull(game.getWinner());

    //every mark in the field is White.
    ((OthelloGame) game).board.setField( i: 28, Mark.WHITE);
    ((OthelloGame) game).board.setField( i: 35, Mark.WHITE);
    ((OthelloGame) game).switchTurn();
    assertEquals(game.getWinner(),game.getTurn());

    //board is full of mark. and the number of black is more than white.
    for (int i=0;i<33;i++){
        ((OthelloGame) game).board.setField(i,Mark.BLACK);
    }
    for (int i=33;i<64;i++){
        ((OthelloGame) game).board.setField(i,Mark.WHITE);
    }
    assertEquals(game.getWinner(),player1);

    //number of mark is exactly same.
    for (int i=0;i<32;i++){
        ((OthelloGame) game).board.setField(i,Mark.BLACK);
    }
    for (int i=32;i<64;i++){
        ((OthelloGame) game).board.setField(i,Mark.WHITE);
    }
    assertNull(game.getWinner());
```

- First the test sets all the pieces on the board to be black and checks if the winner is in fact black (player1) with game.getTurn()
- Next two expressions set the board to the starting position
- Then it checks if the getWinner() returns null in that case
- Afterwards it sets the whole board to white, switches turns and checks if white (player2) is in fact the winner
- Next the board is set to be full with 1 more black piece and checks if It detects black as the winner
- Lastly the board is set to be full with an equal amount of both
- The test checks if the return of getWinner is null in that scenario since there is no winner

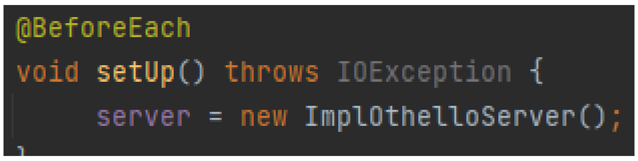


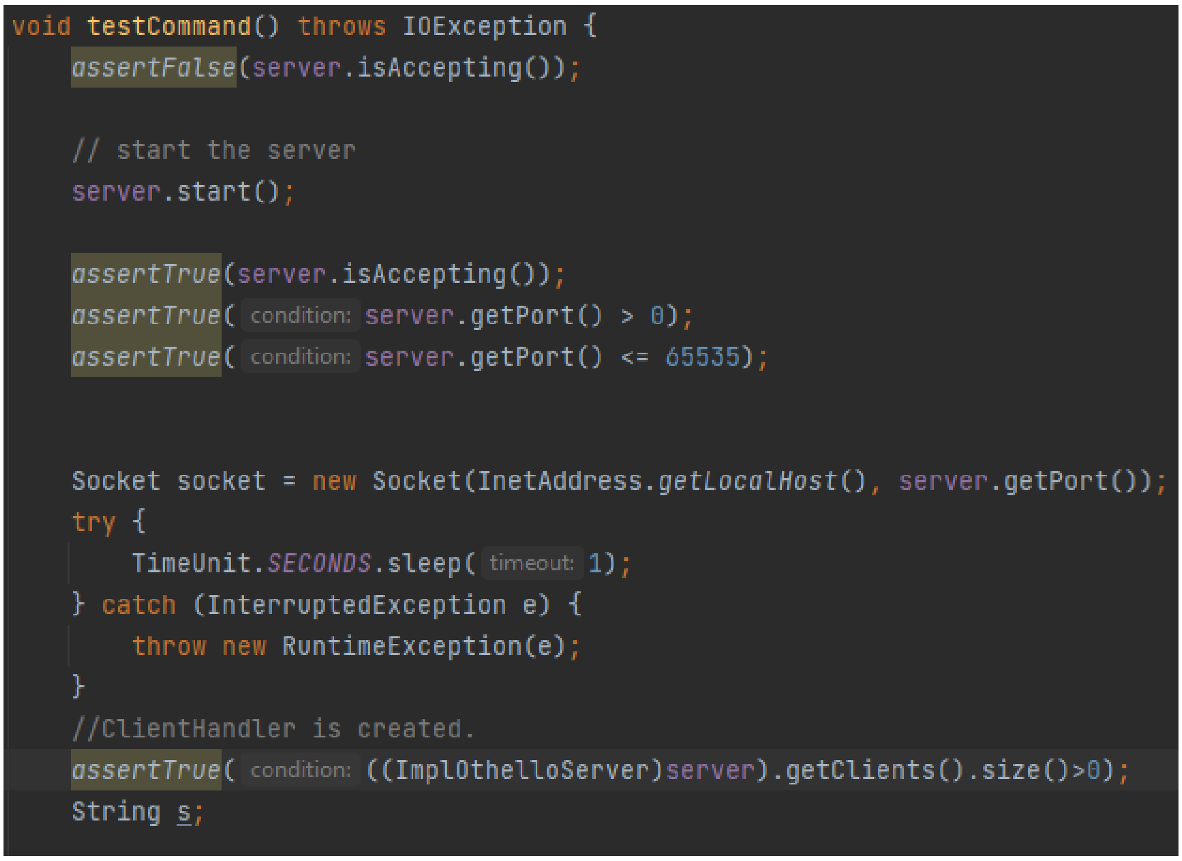
```
@Test
public void TestValidMove(){
    //as default the number of valid moves must be 4 for black.
    List<Move> move = game.getValidMoves();
    assertEquals(move.size(), actual: 4);
    ((OthelloGame)game).switchTurn();
    //as default the number of valid moves must be 4 for white.
    move = game.getValidMoves();
    assertEquals(move.size(), actual: 4);
}
```

- First the test checks if the number of valid moves returned by getValidMoves is in fact 4 at the beginning of the game for black
- Then switch turn and check for the same for white

d) ServerTest

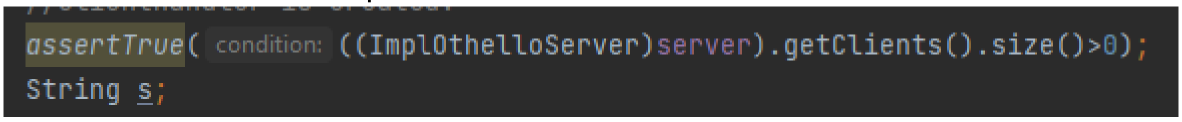
Testing Report for Server
The game Server should be able to connect and communicate with any client using the proper protocol. The client should be able to request and send data to the server using predefined commands and following the protocol. In the case of bad protocol the server should not crash.
Expected behavior: - Before starting the server it should not be accepting any connections - After starting it should be accepting connections at any port number between 0 and 65535 - Connecting to the server socket should automatically start a new Client Handler and store it in the Clients list - Sending a proper HELLO message with a username to the server should prompt the server to respond with the same - Sending a proper LOGIN message with a username should prompt the server to login a client with that username in the server and respond with the same - Sending a proper LIST message with a should prompt the server to respond with a list of all the currently logged in users in the server - Sending a proper QUEUE message should not return any errors from the server side

Actual behavior:

`@BeforeEach`
`void setUp() throws IOException {`
`    server = new ImplOthelloServer();`
- The test setup simply creates a new server



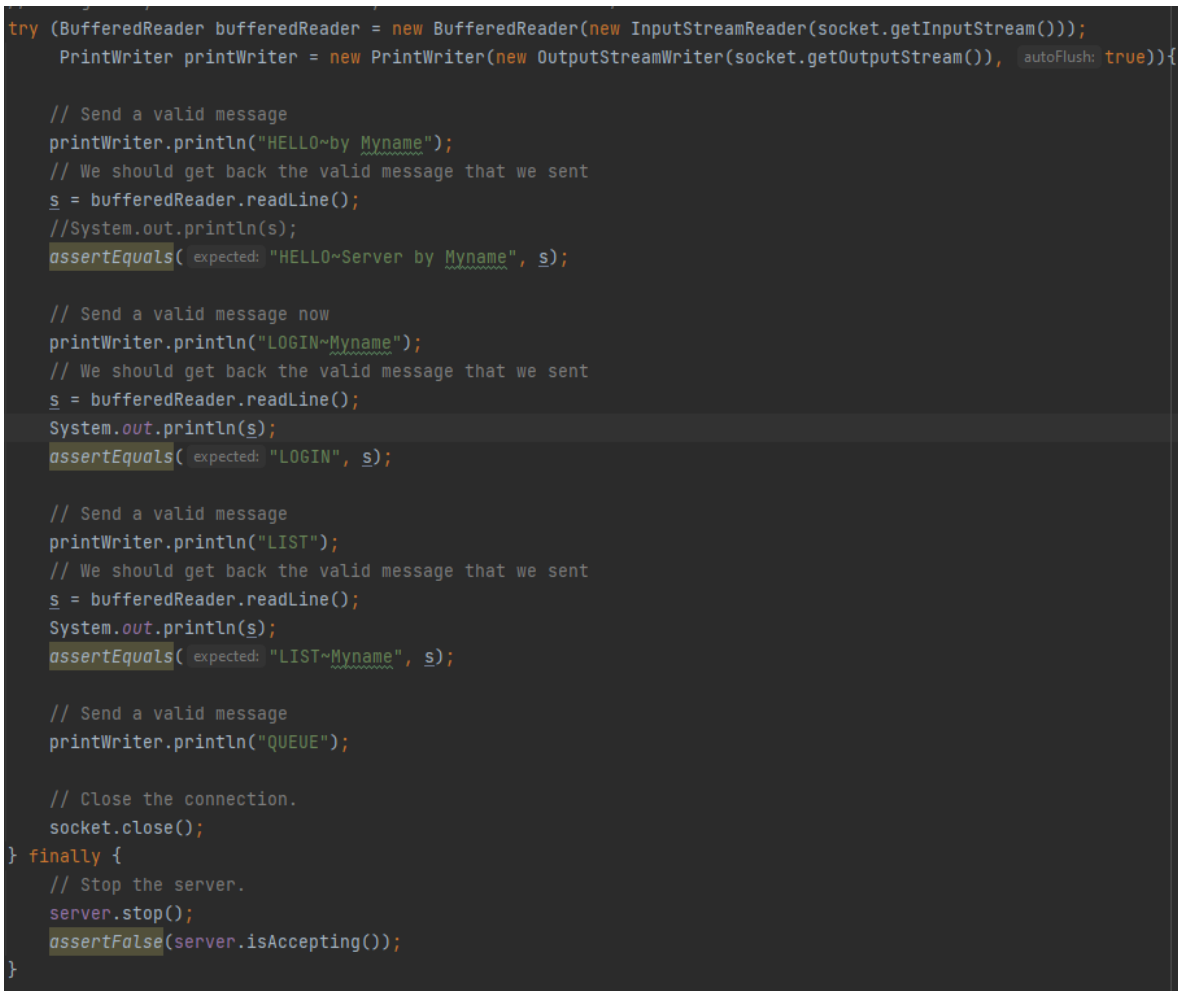
```java
void testCommand() throws IOException {
    assertFalse(server.isAccepting());

    // start the server
    server.start();

    assertTrue(server.isAccepting());
    assertTrue( condition: server.getPort() > 0);
    assertTrue( condition: server.getPort() <= 65535);


    Socket socket = new Socket(InetAddress.getLocalHost(), server.getPort());
    try {
        TimeUnit.SECONDS.sleep( timeout: 1);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    //ClientHandler is created.
    assertTrue( condition: ((ImplOthelloServer)server).getClients().size()>0);
    String s;
```

- Firstly the test checks if the server is not accepting connections when it hasn't been started yet
- Then the server is started and the test checks if it does accept and if the set port number is between 0 and 65535
- Next a socket is setup and connected to the server



```java
assertTrue( condition: ((ImplOthelloServer)server).getClients().size()>0);
String s;
```

- The test checks if connecting with the socket automatically created a new ClientHandler in the server



```
try (BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
     PrintWriter printWriter = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()),  autoFlush: true)){

    // Send a valid message
    printWriter.println("HELLO~by Myname");
    // We should get back the valid message that we sent
    s = bufferedReader.readLine();
    //System.out.println(s);
    assertEquals( expected: "HELLO~Server by Myname", s);

    // Send a valid message now
    printWriter.println("LOGIN~Myname");
    // We should get back the valid message that we sent
    s = bufferedReader.readLine();
    System.out.println(s);
    assertEquals( expected: "LOGIN", s);

    // Send a valid message
    printWriter.println("LIST");
    // We should get back the valid message that we sent
    s = bufferedReader.readLine();
    System.out.println(s);
    assertEquals( expected: "LIST~Myname", s);

    // Send a valid message
    printWriter.println("QUEUE");

    // Close the connection.
    socket.close();
} finally {
    // Stop the server.
    server.stop();
    assertFalse(server.isAccepting());
}
```

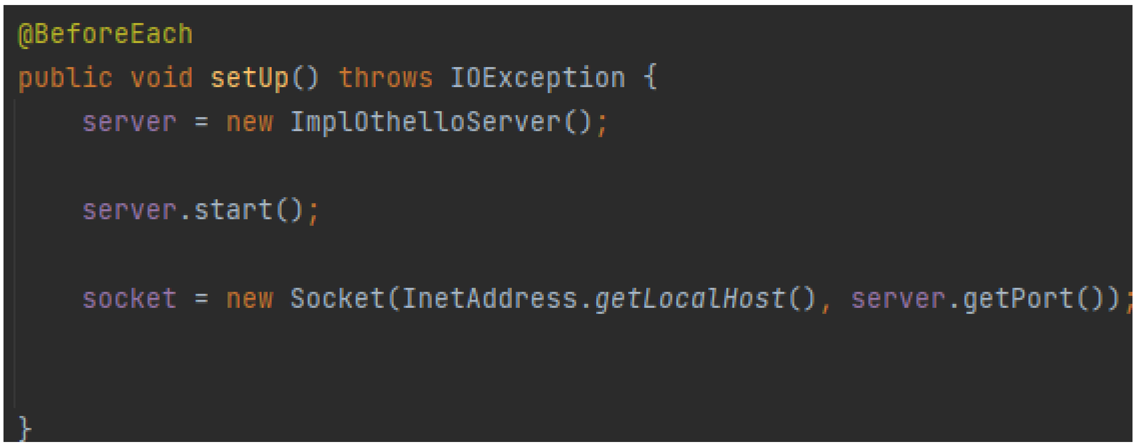
- In this block, a command with proper protocol is being sent to the server through the created socket
- After each command a buffered reader reads from the output stream of the server, prints the messages and checks if each of the responses is what we expect
- For example a “LIST” query should return “LIST~”+ the name of all logged in people which in this case is “Myname”

e) ProtocolTest

Testing Report for Protocol

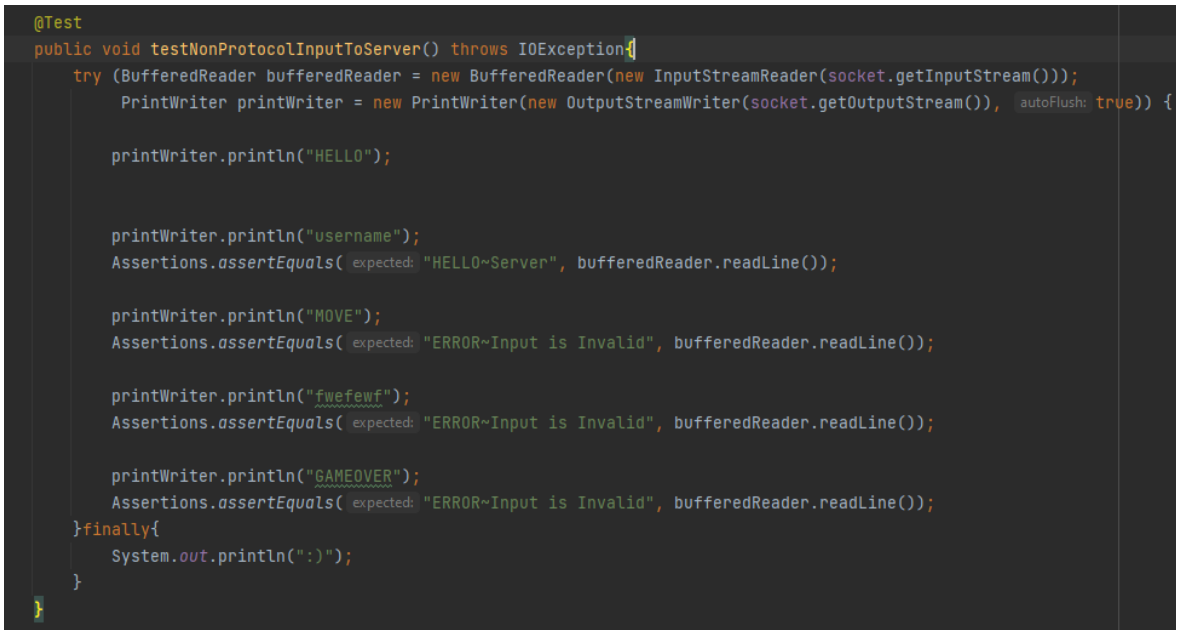
The system implementation should only function when following the defined protocol. In the case one of the parties (either client or server) is not adhering to it, our system should detect the error but not break or crash at any circumstance.

Expected behavior:
- When a command gets sent to the server in different order than intended as the server shouldn’t crash but acknowledge the error and send appropriate message
- When a message that doesn’t follow protocol gets sent to the server, it shouldn’t crash but acknowledge the error and send appropriate message



```
@BeforeEach
public void setUp() throws IOException {
    server = new ImplOthelloServer();

    server.start();

    socket = new Socket(InetAddress.getLocalHost(), server.getPort());


}
```

- Setup creates a new server, starts it and connects to its socket with a client handler.



```
@Test
public void testNonProtocolInputToServer() throws IOException{
    try (BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
         PrintWriter printWriter = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()), autoFlush: true)) {

        printWriter.println("HELLO");


        printWriter.println("username");
        Assertions.assertEquals( expected: "HELLO~Server", bufferedReader.readLine());

        printWriter.println("MOVE");
        Assertions.assertEquals( expected: "ERROR~Input is Invalid", bufferedReader.readLine());

        printWriter.println("fwefewf");
        Assertions.assertEquals( expected: "ERROR~Input is Invalid", bufferedReader.readLine());

        printWriter.println("GAMEOVER");
        Assertions.assertEquals( expected: "ERROR~Input is Invalid", bufferedReader.readLine());
    }finally{
        System.out.println(":)");
    }
}
```

- First send necessary commands for the server to recognize you as use
- The the test tries sending different kinds of commands that don't adhere to protocol
- The buffered reader reads output of the server
- Each assertEquals checks for the same output, that the input is invalid

Reflection on process

Shun

In terms of the process of the whole project, we should have made a checkpoint for tracking the coding process, otherwise, we have to hurry up before the deadline.

Ideally, we’d better make a whole class diagram and sequence diagram to check what we need to develop. Moreover, at the beginning, we should have a concrete combined schedule between Test schedule and the coding schedule.
Also, we needed to frequently check the target of functionality and how the final development works, because we lost much time to develop functions which do not suit the server and clients given by the university as well as Protocol.
Apart from scheduling and the process of doing development and design, all coworking was really comfortable. Sometimes we divided tasks to finish them quickly. And also, especially coding tasks, we work together using pair programming. So, totally our combination was successful and effective.

Mat

I really liked working with someone on a very involved project; it was unlike anything I’ve ever done before. I believe we gave it our best shot and it made for an interesting learning performance. That being said, I believe better time management and more realistic expectations when it comes to completing certain stages of the project could have really helped to keep the working process more structured. This is a tough challenge since neither of us were ever a part of a project this big so we had no idea of just how time consuming would make all the parts of the system work together while eliminating all the possible errors.