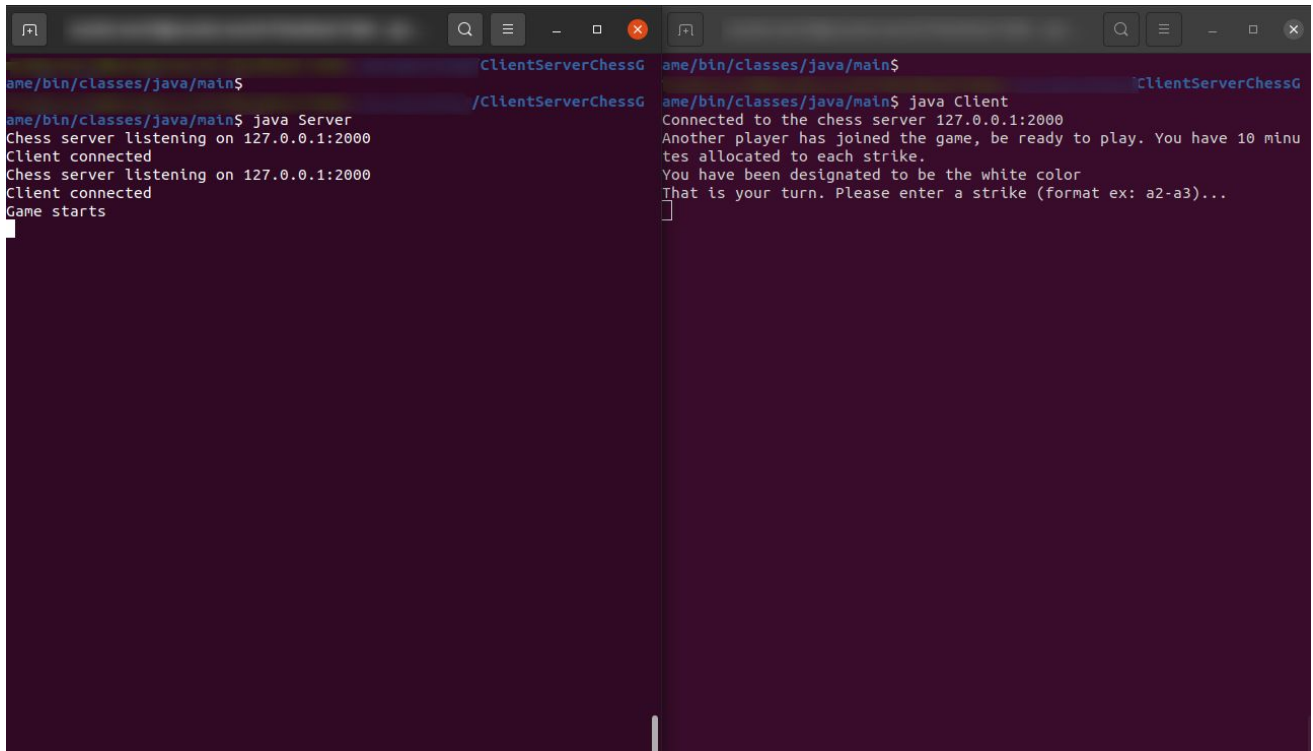


Client/Server Chess Game : User Guide

Firstly and foremost, compile and run the project as described in the *README.md* file.

The project has been done in accordance with the specification document, except that we agreed to directly implement a 2-player server, and I implemented a GUI when a player request the board to the server (command *display_board*). A few notes therefore need to be done.

Here is what is shown when 2 players connect - the screenshots include only one client for readability.



```
ame/bin/classes/java/main$ java Server
Chess server listening on 127.0.0.1:2000
Client connected
Chess server listening on 127.0.0.1:2000
Client connected
Game starts

ame/bin/classes/java/main$ java Client
Connected to the chess server 127.0.0.1:2000
Another player has joined the game, be ready to play. You have 10 minutes allocated to each strike.
You have been designated to be the white color
That is your turn. Please enter a strike (format ex: a2-a3)...
█
```

Server on the left and a client on the right

Then the player for whom that is his/her turn can enter a mover according to the specification document.

```
ame/bin/classes/java/main$ java Client
Connected to the chess server 127.0.0.1:2000
Waiting for another player... 1 minute timeout
Another player has joined the game, be ready to play. You have 10 minutes allocated to each strike.
You have been designated to be the black color
Waiting for your opponent to play... (any command except exit will be ignored)
1. white pawn moves from a2 to a3.
That is your turn. Please enter a strike (format ex: a2-a3)...
█

ame/bin/classes/java/main$ java Client
Connected to the chess server 127.0.0.1:2000
Another player has joined the game, be ready to play. You have 10 minutes allocated to each strike.
You have been designated to be the white color
That is your turn. Please enter a strike (format ex: a2-a3)...
a2-a3
1. white pawn moves from a2 to a3.
Waiting for your opponent to play... (any command except exit will be ignored)
█
```

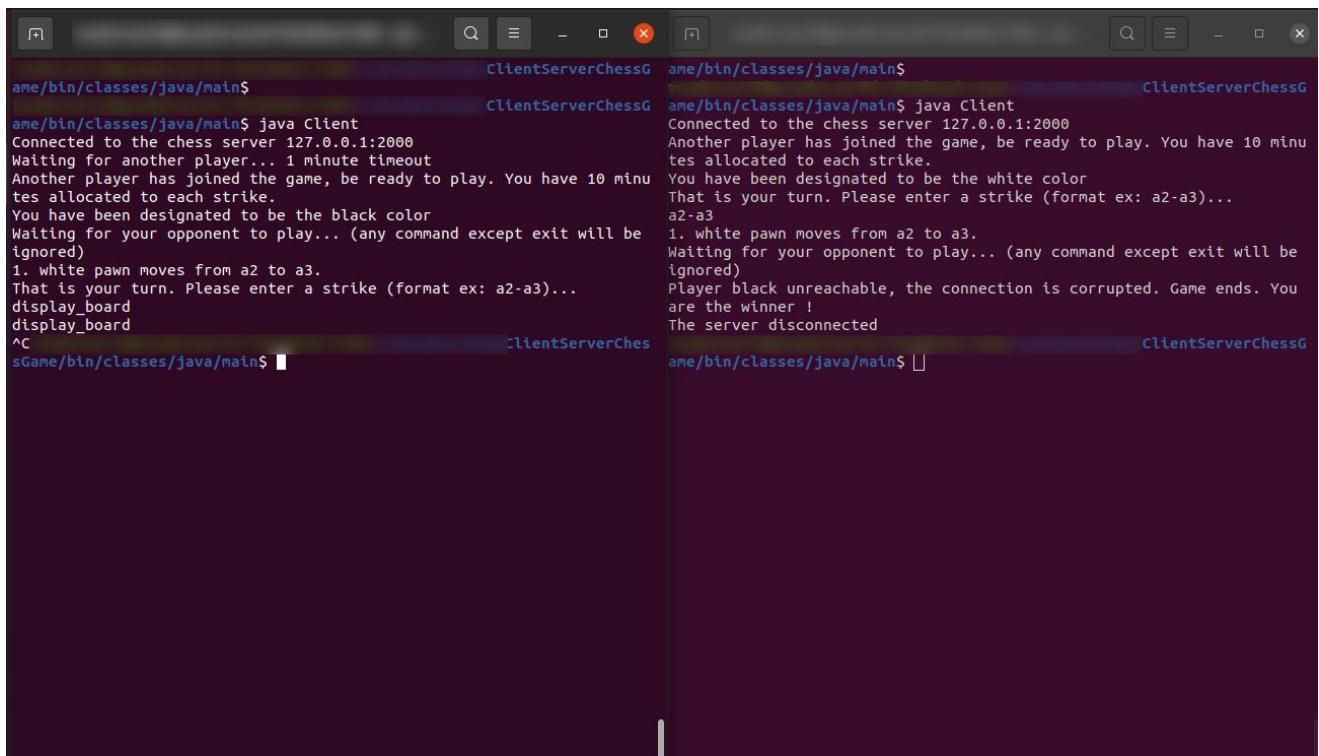
Server replaced by the other client on the left, and same client on the right

A GUI is displayed if the player wants to display the board.



The two clients as before

The code handles the case of unexpected behaviours, for example the disconnection of a player.



```
ame/bin/classes/java/main$ java Client
Connected to the chess server 127.0.0.1:2000
Waiting for another player... 1 minute timeout
Another player has joined the game, be ready to play. You have 10 minutes allocated to each strike.
You have been designated to be the black color
Waiting for your opponent to play... (any command except exit will be ignored)
1. white pawn moves from a2 to a3.
That is your turn. Please enter a strike (format ex: a2-a3)...
display_board
display_board
^C
ame/bin/classes/java/main$
```

```
ame/bin/classes/java/main$ java Client
Connected to the chess server 127.0.0.1:2000
Another player has joined the game, be ready to play. You have 10 minutes allocated to each strike.
You have been designated to be the white color
That is your turn. Please enter a strike (format ex: a2-a3)...
a2-a3
1. white pawn moves from a2 to a3.
Waiting for your opponent to play... (any command except exit will be ignored)
Player black unreachable, the connection is corrupted. Game ends. You are the winner !
The server disconnected
ame/bin/classes/java/main$
```

The left player left the game

A few notes are still required to be done.

- The color of the player is set randomly.
- Timeouts are set for the connection of the players and for a strike to be received from a client.
- The maximum of cases has been treated, only 2 are not - the players can agree to a draw, and a game is claimed a draw when there is not enough piece to make a checkmate. This point will be discussed further during next interview.
- The ip/port options are available.
- The client file option is available. As we agreed an incorrect move causes the player to be disconnected. When the file is fully read, the program switch to command line. Given the first point, the file must contain the strikes of both players, and the program reads the appropriate lines. Ultimately, the program may work with 2 different files, but it does not really make sense.
- The verbose option is not yet treated. Note that a lot are already displayed, and the verbose mode could control that. In addition, some functions are already in code for when the verbose mode will be taken into account. The verbose mode could also include some of the logs, and I aimed to include information on an rejected move ("Invalid move" from the server) for the verbose.
- The logger has not been implemented yet.
- Manual tests have been done to make sure the game is working appropriately. Yet I didn't got the time to make automated tests (unit tests etc) by using *JUnit* and *Google c++* testing framework.