

**SOFE4790 Distributed Systems**

**(Fall 2020 - Dr. Q)**

**Assignment #1 – Multi-threaded Connect Four Server / Client**

**Honour code**: By submitting this assignment report, I (name and banner ID# below) confirm this is my own work, and I have not asked any of my fellow students or others for their source code or solutions to complete this assignment, and I have not offered my source code or solutions for this assignment to any of my fellow students.

**Name**: Daniel Nucci  
**Banner ID#**: 100655384

Introduction:

For my multi-threaded server / client program I created a Connect 4 board game in the command line. The client is built to have as little code and memory usage as possible and push all the ‘rendering’ and processing to the server. In the case of this Connect 4 program in particular, the client only deals with approx. 1 to 2 characters, whereas the server keeps track of 2 Connections per GameBoard per Match and all associated data. ‘MTServer.java’, ‘ConnectFourClient.java’, and ‘README.txt’ are the associated java files for this assignment, documented and compiled into .class files in the Github Repo (<https://github.com/UOITEngineering2/assignment1fall2020-djnucci>). Instructions for installation and running the program are in the README.

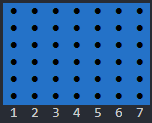
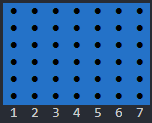
Application:

This program is an exercise into if threads work well with each other. The main server can handle many red-coloured chip players in an open lobby waiting to connect with their associated yellow-coloured chip players. For a match to be set up a red and a yellow player must be present to place pieces on the board. Once both players connect, they play a match of Connect 4. The winner is then displayed to them once the match is over. As this program is not to be useful the application is to just have fun with the game and having fun programming the game.

Novel Features / Functionalities:

I have a lot of features in this program that I would possibly classify as novel, I’ll list them below with associated java files and line numbers / methods.

* Novel use of ANSI characters to display Connect 4 board on Client terminal
  + MTServer.java → Connect4Board → (visualiseChips(Boolean isNotSimpleConnection) Line:136)
  + **Simple text mode** can also be enabled for people with colour blindness or an outdated terminal
* Novel Implementation of victory checking
  + MTServer.java → Connect4Board → (getChipVictoryState(int chipColour, int posX, int posY)) Line:225)



(a)

(a)

Fig. 1. (a in red) highlighting the only chips that need to be checked for victory state, (b in purple) highlighting all lines needed to be checked

* + It was observed that every single victory must pass through these two lines. As this is the case these chips were the only ones that needed to be checked on the board. As each chip is checked, a counter keeps track of how many chips of the same colour are in the same line together. If the number of chips is above 4, the methods returns the colour of the victor.
* Optional computer-controlled player
  + MTServer.java → ConnectFourAIClient → (Line: 793)
  + This is not a particularly sophisticated AI in a bot player, but the purpose of the assignment was to get multi-threading in a server to work, not a bot that can beat a human.
* Command line flag variables for the server and the client to easily set variables without dialog
  + MTServer.java → main(String args[]) → (Line:23)
  + MTServer.java → main(String args[]) → (Line:10)

Diagram:

A simple view of the objects will now be shown.

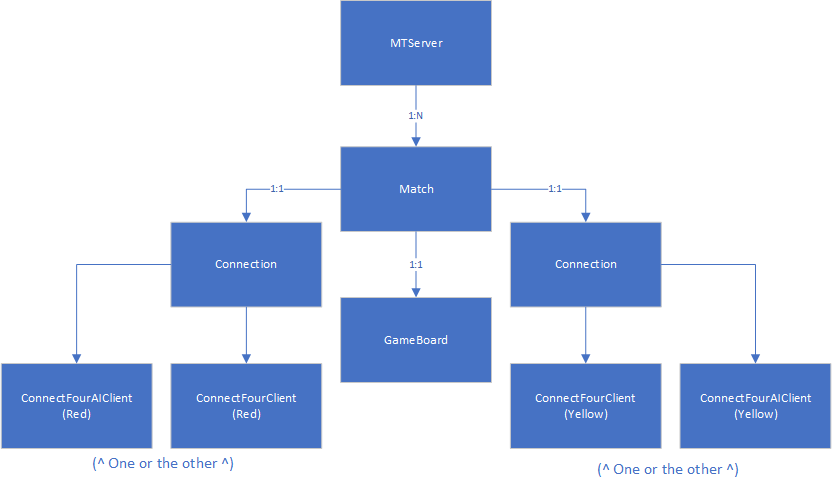


Fig. 2. Each Server can have many Matches, each match can only have 2 Connections, and each Connection can be filled with a Bot or a Human

Challenges and Solutions:

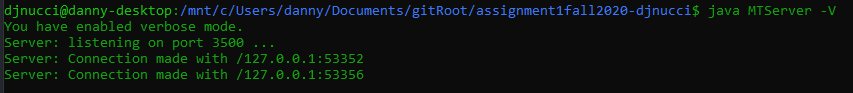
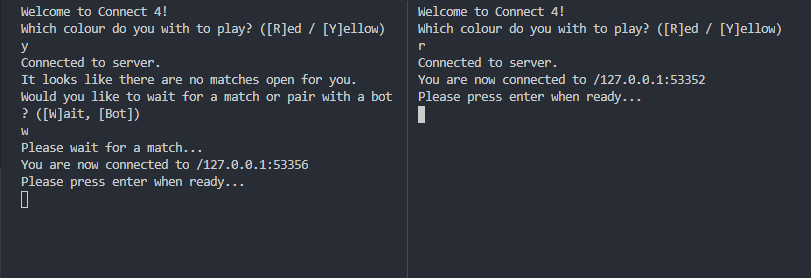
The main challenge of this assignment was to make sure the two clients synced up their match play. This was solved by a careful note of what is being sent to who and when. Other than that, there were no challenges.

Tests:

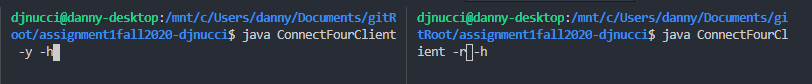
Compilation of the program:

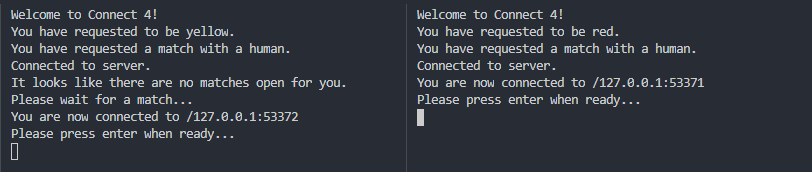


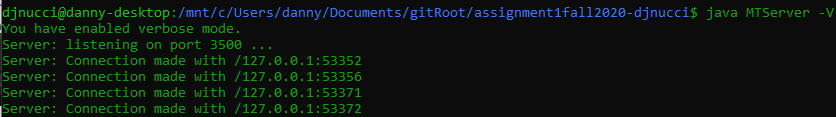
Connecting to a Match in many ways:



These users connected to each other through dialogs (and the server taking note)

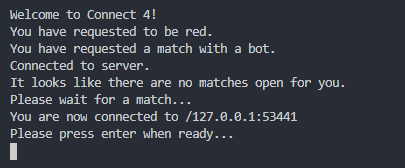


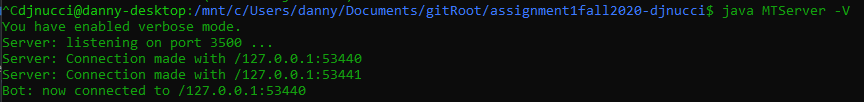




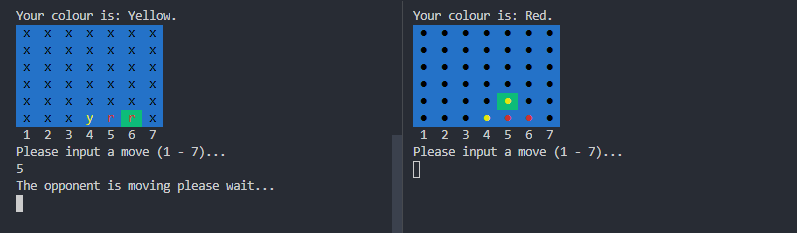
These users connected with args in command line (and the server taking note)

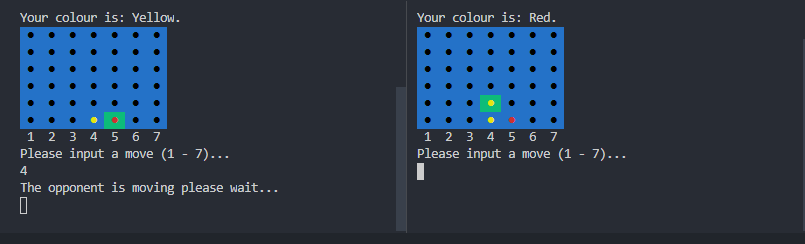




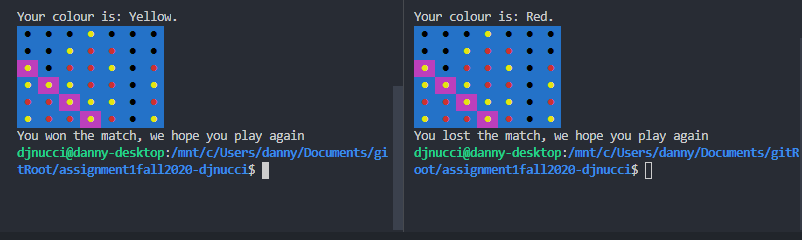


This user wants to play a bot (and the server takes note)

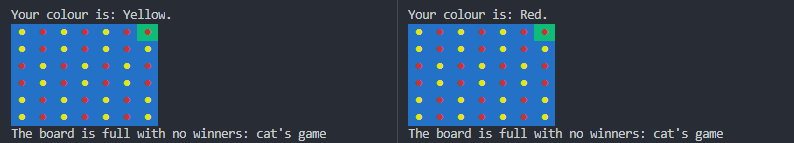
  
the -s flag is on a per client basis; therefore one client can be connected with simple text and the other with ANSI symbols

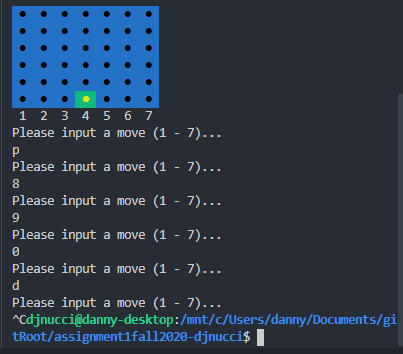


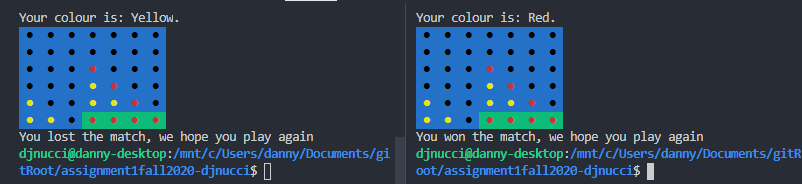
Gameboard functionality (the last placed piece is highlighted)

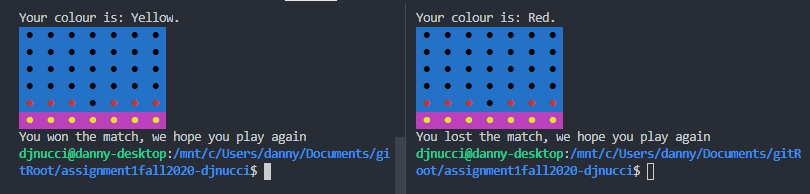


The Yellow player is victorious (the highlight is purple for a yellow victory condition)

  
On a Cat’s game the final move is still highlighted to show where the last chip was placed

  
Example of one of many improper formatting handling (all client input is error handled)

  
In the case of multiple victories happening at the same time,   
the first the computer sees is the highlighted victory

  
All chips in the victory line will be highlighted

Server running with all or no flag options:



