DATA COMMUNICATIONS AND NETWORKING

PROJECT

WRITING TCP CLIENT AND SERVER SOFTWARE - 3 STONES

C. Davis

A. Simonelis

Due/demos: October 16, 2019

Docs: October 18, 2019

You will work in assigned groups. Each group will demonstrate its work, as specified by your instructor. All group members must be present at the demonstration. All members of a group are expected to contribute more or less equally to the project, and to be familiar with all of the work of the project.

Each team will write a client-server TCP-based system in Java that will allow a person using the client to play game(s) of 3 Stones with the computer server.

At initiation, the server software will display its IP address. The client will allow the user to input the IP address of the server. The machines will then connect to begin the session, after which:

-the client program will allow the user to start a game, or to quit the client; the client will allow the user the same two choices at the end of each game; it is desirable, but not essential, for the client to support a feature that allows the user to quit the client at any time, via the mouse or the keyboard;

- the server will continue servicing the client that began the session until the client terminates the session by choosing not to play another game (or by closing the socket), at which point the server will resume listening for new clients (i.e. the server will be an iterative server, servicing one client at a time).

Both the client and the server are to display appropriate messages while running. The client will display the game board while a game is in progress. Your display may either use a GUI or be text based.

Your 3S protocol will have the server listening on port 50,000.

All 3S messages must be fixed-length, multibyte packets. Design your packet layout carefully.

A 3S session consists of zero or more games.

The client initiates a game by sending the server a request-game-start message (e.g. 0x00000000, if 4-byte packets are being used). The server replies with a game-start message (e.g. 0x0000000A), after which the client draws the board, . . .

Subsequently, the game is played as follows. 3 Stones is a strategy board game akin to tic-tac-toe. Your version of this game will have the client play the first stone onto the modified board (see bottom of document), and then the server and client will alternate playing stones. Each will try to score more points than the opponent by the end of the game, which occurs when all stones have been used. The client starts with 15 white stones and the server starts with 15 black stones. At the end of the game the client must display a message showing the 2 scores and who the winner is, or whether a draw occurred. The server is responsible for all of the scoring logic.

Additional rules:

* a stone placement scores 1 point for EVERY three-stones-in-a-line that it completes, vertical and/or horizontal and/or diagonal
* the first stone may be played in any slot
* any other stone must be played into an empty slot that is either in the same row or the same column as the last stone played
* if there are empty slots neither in the same row nor in the same column as the last stone played, the next stone may be played in any empty slot
* slots on opposite sides of the center are considered to be part of the same row or column for the purposes of next-stone placement; however, three-stones-in-a-line cannot be scored through the centre.

Features:

* visually highlight the last stone played
* consider using an 11 x 11 array to represent the board
* do not expect your server to play a perfect game; you may use brute force logic to examine all possibilities for the server’s stone placement, evaluate the point score of each, and then pick the placement that gets the server the most points
* for a more intelligent server, in addition to the above point, examine the same possibilities but evaluate how many white points are blocked by the placement, and then pick the placement that has maximum(black-points-scored, white-points-blocked) points; if the maximum is 0, pick a random empty slot, or an empty slot beside one of the server’s played stones, or an empty slot at a ‘good’ position on the board, or . . .
* consider other intelligent algorithms
* if the server may play in any empty slot, the simplest choice is to search for the first free slot or a random empty slot; a more intelligent choice would be to pick a free slot near an existing server stone, or . . .

When programming 3 Stones, use data structures and algorithms that are as simple and clear as possible.

Include good, explanatory Javadoc comments.

You must follow the principles of object oriented design, and have an up-to-date UML class association diagram available for inspection by your instructor whenever you are working on the project.

You must use GitLab to create and manage the project. If you don’t have a GitLab account, you are required to create one, and use this account to contribute to the project. One member of each group will create a group for the project. **The permission of each group must be set to Private,** and no one should be allowed to access the project, except the team members and your instructor. All the team members and your instructor (he will provide you his user name) must be added to the group that was created for the project. The user permission of each member of the group (including that of your instructor) must be set to Maintainer.

**All lines of code in this project must be your own.**

Final documentation (per team) will consist of:

1. A program narrative
2. The class association diagram
3. Source listings of the client program and the server program, both of which are to include good explanatory comments; on the day that you submit your documentation
4. All of the above need to be uploaded to your Git repository

