CS 342 Project#5 Summer 2019

TicTacToe w/ Min-Max

Due Date: Saturday, August 10 2019, @ 11:59pm

Description:

This is your second team project. You will create a server that plays Tic Tac Toe with each client that connects to that server. The server will utilize the Min-Max algorithm to determine its moves as well as when the game has been won or tied. At the end of each game, each client will be able to play again or quit.

Like Project #4, All networking must be done utilizing Java Sockets (as shown in class). The server must run on its own thread and handle each client on a separate thread. You may not use libraries or classes not included in the standard Java 8 release. You may not alter the POM file.

Implementation Details:

The client and server programs for this project will be Maven projects. You can use the Maven templates that were included in Project #3. You will provide the following functionality:

- Each client can see a list of the top 3 highest scores.
- The server will not limit the number of clients connected at one time
- When each client logs on, they will play a game of Tic Tac Toe with the server.
- When that game is complete, they can choose to play again or quit.

To clarify, if there are 6 clients on the server at one time, there would be 6 games on 6 different client threads, playing against the server simultaneously. The client will always be **‘O’** and the server will always be **‘X’**. After each game is complete, the number of client wins should be updated as well as the list of top 3 scores. Each client should see the updated list. Otherwise, Clients do not need to know about each other.

You must still utilize the GameInfo class as the only means of passing information between the server and each client. You must incorporate the Min-Max code provided to implement the servers Min-Max algorithm. More on that below.

The Server GUI must have a start up scene to enter the port and start the server. Also a list view that records the actions and traffic on the server. Feel free to add any other elements you feel necessary. Also feel free to implement the “look and feel” as you see fit.

The client must also have a startup scene that allows the user to enter a port and ip address to connect to the server. Also, the client must have a way to display the current game board, top 3 scores and a “play again” button. Feel free to add any other elements you feel necessary. Also feel free to implement the “look and feel” as you see fit.

The Min-Max Algorithm:

In the included code, the Min-Max algorithm takes in the current state of the Tic Tac Toe game. It assumes that the next move is always **X.** So, you will want to limit your clients to playing **O.** This algorithm goes back and forth between two recursive methods, exploring every possible state of the game for every possible next move by **X** and response by **O**. When it reaches an end state, it will percolate that result up as the recursion unfolds and give **X** a score for each possible current move.

You should run the code with several different states to understand what it is doing!

Your server will incorporate this code in a class called FindNextMove. In that class, you will want to take the current state for a game and call the Min-Max algorithm to inform the server of its next move. The algorithm must run on it’s own thread. Your server will have a single instance of the FindNextMove class that will be used to determine each next move for each client game. You will also need to utilize synchronization to avoid multiple client threads corrupting the results. For instance, Min-Max returns a list of moves but it is a combination of moves from multiple games on different threads.

You may only use the Min-Max code provided. You may alter it if you wish to suit your servers logic. The Min-Max algorithm must run on its own thread and use synchronization to avoid race conditions.

Testing Code:

You are required to include JUnit 5 test cases for the Min-Max code provided. You must thoroughly test this this code to ensure its functionality. Add these to the src/test/java directory of your Maven Project.

CS 342 Project#5 Summer 2019

UML Class and Activity Diagrams:

You are required to include a UML Class Diagram and UML Activity Diagram for the server and client programs; including all classes and interactions between them. Format these as PDFs.

Use of templates found on the web:

You may use ideas from templates(fxml, css or other) for styling your programs found on the web. You may not import those templates and pass them off as your own work. If you use an idea or part of such template, include a reference to that work in the header of the file where used. Failure to do this will result in academic misconduct charges.

Documentation:

You will submit a single page discussing how you incorporated the Min-Max code into your overall design including any changes made to the original code. Also, give a general description of the server and client logic. This description should highlight and discuss algorithms and design choices made. There must be a cover page that has your team number, names and emails of each team member. Format this as a PDF.

Electronic Submission:

Zip the Maven projects P3Server and P3Client together along with the class and activity diagrams and documentation PDF and name it with your netid + Project5: for example, I would have a submission called mhalle5Project5.zip, and submit it to the link on Blackboard course website. Please only submit one project per team.

CS 342 Project#5 Summer 2019

Assignment Details:

Late work is not accepted. Anything submitted late will not be graded and result in a zero.

We will test all projects on the command line using Maven 3.6.1. You may develop in any IDE you chose but make sure your project can be run on the command line using Maven commands. Any project that does not run will result in a zero. If you are unsure about using Maven, come see your TA or Professor.

This is a team project. You may share and communicate at will with your teammates but are forbidden from collaboration with other teams.

Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml.

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing

your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone

else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml.