**The University of West Florida  
Department of Computer Science**

**Advanced Computer Programming Assignment**

**John W. Coffey**

**Program Description: Network Tic-Tac-Toe!!!**

We will design a program that enables people to play a game of Tic Tac Toe over the Internet. You will use localhost, as described in class, to simplify grading.

**Running the program**

The user will start the server program from a command prompt. The server will alert that it is ready to accept players for the game. The user will then start two clients from different command prompts. The client program will be a graphical interface using JavaFX. When the client starts, it will ask the user for his or her name and then it will send that name to the server.

The server will create games as pairs of clients make requests. Once each game has two players, the server will accept move commands with three parameters: the player number (1 or 2), the row of the Tic Tac Toe board (0, 1, or 2) and the column of the board (0, 1, 2). Moves must alternate between players. and each move must be displayed back to both clients. Significant error checking can take place on the clients, but the server should control the board in such a way as to guarantee no race conditions occur. Game play will continue until either a player wins or all board positions have been taken and a draw is declared.

A summary of our Network Tic-Tac-Toe protocol (N3TP) is:

|  |  |  |
| --- | --- | --- |
| **Client Message** | **Server Response** | **Meaning** |
| join <name> | Hello <name> you are player 1  or  Hello <name> you are player 2. Let the game begin! | First player joining the game    Second player has executed a "join" command |
| choose <int> <int> <int> | We do not have two players yet      or  Player <int> has chosen <int> <int>      or  Position <int> <int> is taken, try again      or  Not Player <int> turn      or  Player <int> WINS!!!      or  Illegal player number      or  Illegal Board Position | Need a second player to start    Legal move - not the end of the game    Position has been selected    Must strictly alternate turns    Winner    Bad input data    Bad input data |
| quit | Player <int> has quit | Game over |

**Notes:**

* The goal is to establish the ability to manage multiple games by creating games as pairs of clients join.
* The program must implement several checks:
  + Players must strictly alternate making moves - the program must prevent players from moving twice in a row
  + The program must check that the current move has not yet been made and that the current move is within the confines of the game.
* Every time the board is updated, output must be written to both clients simultaneously. This feature means that the game should interact with a given client until that client has made a legal move, THEN communicate the new board state to both clients.
* Some of these features are more difficult to implement than others. I will provide guidance as needed.

**Deliverables**

DO NOT include .class files. The program must be recompiled before being tested to be sure the .class files are compatible with the runtime environment I use. Otherwise,you will submit all the usual things:   
1. UML class diagram.

2. A User's manual for your program *which tells me how to run it.*  
3. Your source code in Java

4. A batch file (rc.bat) that will run the client and one (rs.bat) to run the server ON WINDOWS. NO Linux.

Please review the policy on Academic dishonesty.

**Submission requirements:**  
1. Compile and run your program one last time before submitting it. Your program must run with JDK 8.0.  
2. Place ALL files in your submission into a folder named <lastName><firstInitial>-p<number>. If I were submitting project 4, my name would be coffeyj-p4.  
3. zip that folder into a .zip file with the SAME NAME. This means that inside your zip file, you will have exactly one folder (from the example: coffeyj-p4) showing at the top level.  
4. DO NOT make separate folders for documentation and source files (or anything else) inside the main folder. Having such a setup simply necessitates more navigation time to get where we need to go to grade.  
5. Any needed input files should be in the top-level folder along with the source code. This program will not need any input files.  
6. MacOS users - *remove the \_MacOS\_ utility folder* before you zip up the file. If you cannot, delete it from the archive once the archive is created. It just takes up space and is not needed for anything we do with your submissions.  
7. Login to Canvas. Select our course from the dashboard.  
9. Select Assignments. Then select the appropriate project.  
10. Upload your zip file. Check to insure that the file was uploaded. ALWAYS give yourself enough time. If you are trying to submit at 11:57pm on your machine, the clock might be off and the dropbox might already be closed.