CSCD 467/567 Final Project

**Specifics**

1. This project is offered to substitute for the final exam. If you plan to do this project, you are encouraged to sign up on a final project sign-up sheet. You can choose either to take the final exam or to work on this final project.
2. This project is team-based and collaborative and can be completed in a team that consists of NO more than two students. Each team has to complete all specified tasks and implement the functionalities described below.
3. Please turn in all your Java source files on Canvas by the specified deadline, each team member has to turn in all your own code. Please organized all your source files in a way such that the grader can compile all your code located in one folder (no subdirectories please) using a command **javac \*.java** and run your code using **java Tester** on command line. On top of your Tester.java file, please explicitly list all team members’ names in a comment section.
4. Students who choose to do the project will have a 9-minute demonstration or presentation at the end of the quarter. Each team has to demonstrate a functional system and how your implementation meets the requirements specified by the instructor.

**Problem Statement**

In this project, you will have to implement a cloud-based TicTacToe (also called ConnectFour) game application. In a start-up package, the instructor has provided the code that implements a client player’s Graphical User Interface(GUI) for the game. The instructor will discuss the start-up code in classroom.

You will implement a system illustrated in the diagram below.

Client Player 1 on computer A

Using mark symbol X

Client Player 2 on computer B

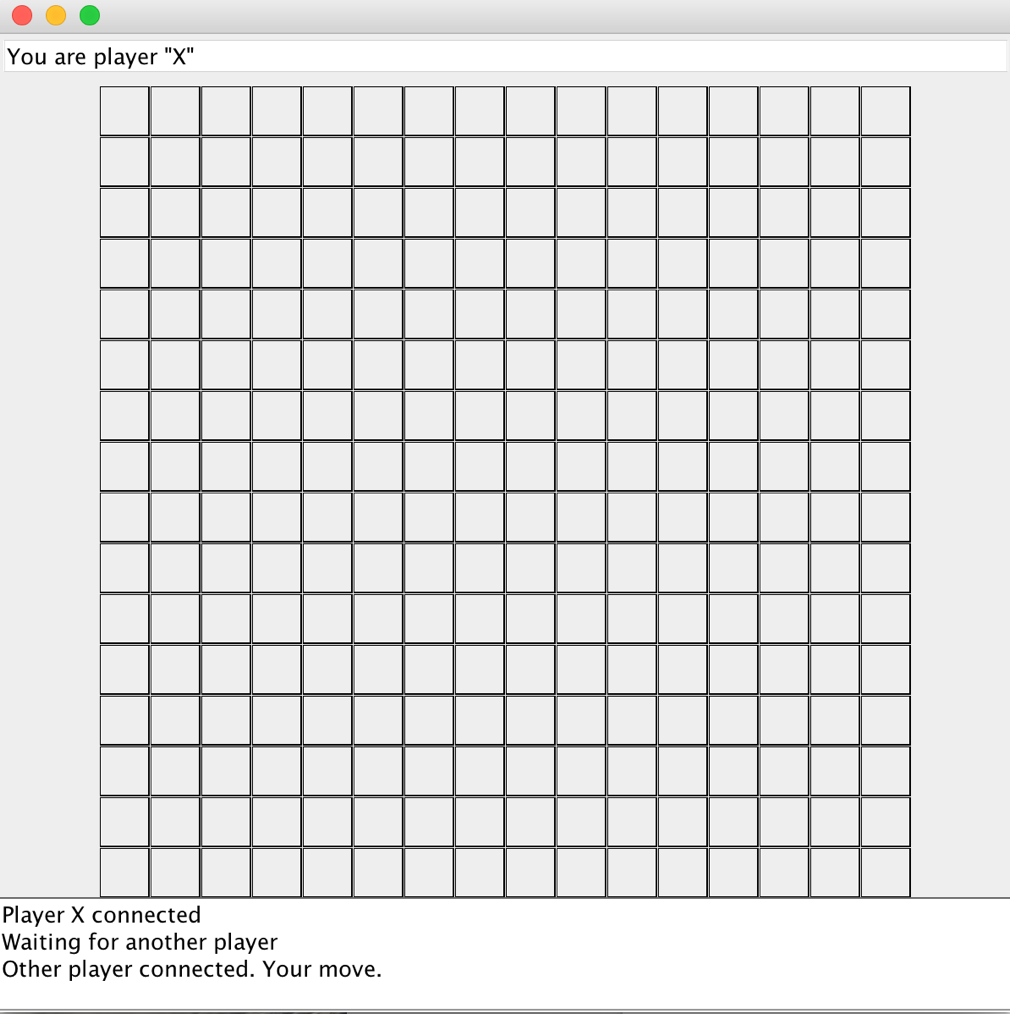
Using mark symbol O.

In the diagram above, your program allows two players to play the TicTacToe game on two different computers which communicate to each other using one or more of the AWS Cloud services, such as S3, DynamoDB, SQS, SimpleDB, Lambda, SNS, or IoT. You are not allowed to use an EC2 virtual machine for the communication purpose.

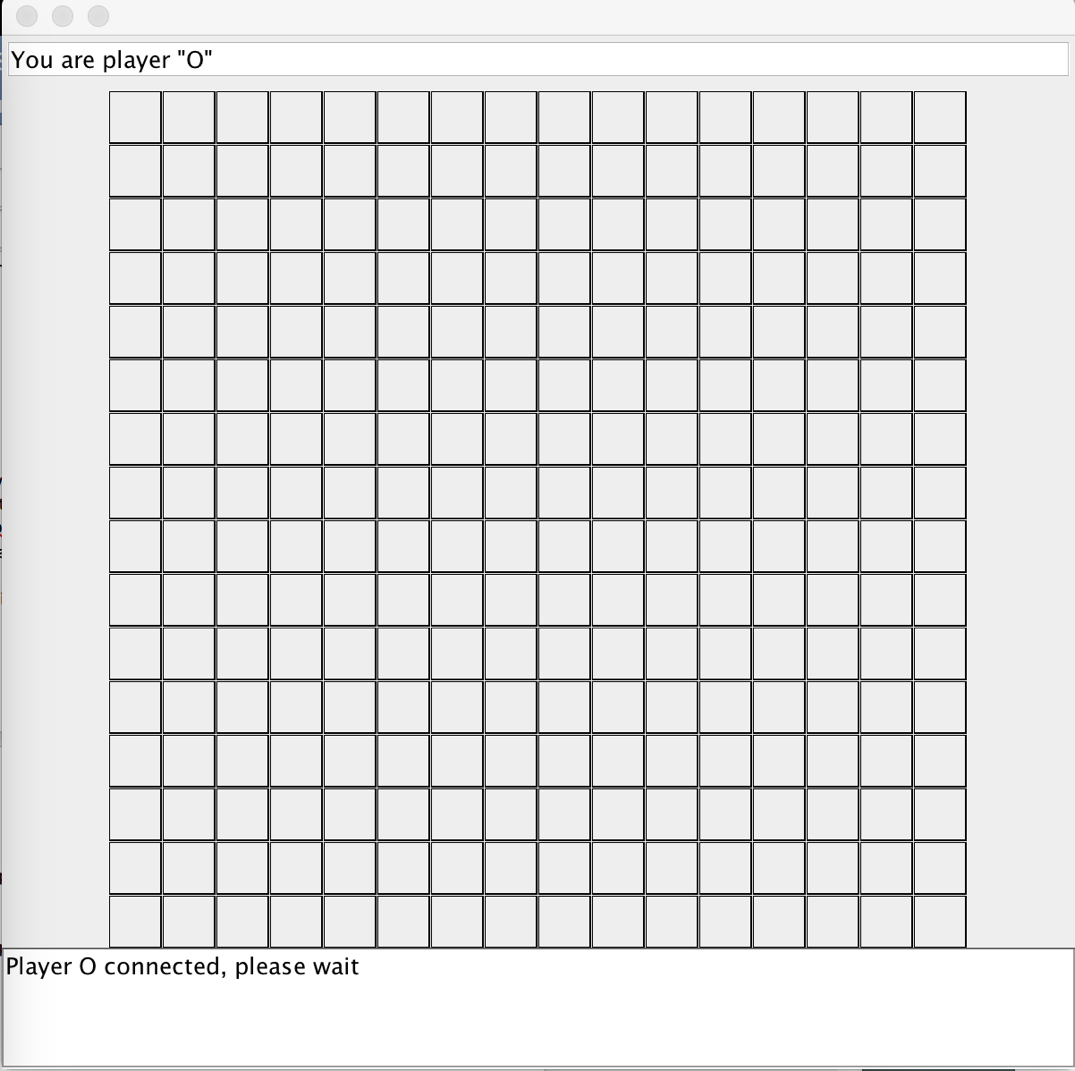
If your team is made up of two members, such team has to implement two different designs **in which different set of AWS Cloud services are used**. For example, in the first design your team may use AWS SimpleDB for synchronization and communication, while in the second design your team may use AWS SNS and Lambda for communication. So that each member in a team can work on a different design and can learn from each other. In your submission, on top of your Tester.java file, please explicitly describe which AWS service(s) you have used in your code. Each team member in a team is required to submit a **different** design and implementations. If your team has only one member (working on your own), the student is allowed to implement only one design, instead of two designs.

**In addition to the general requirements described above, you are required to implement the following features.**

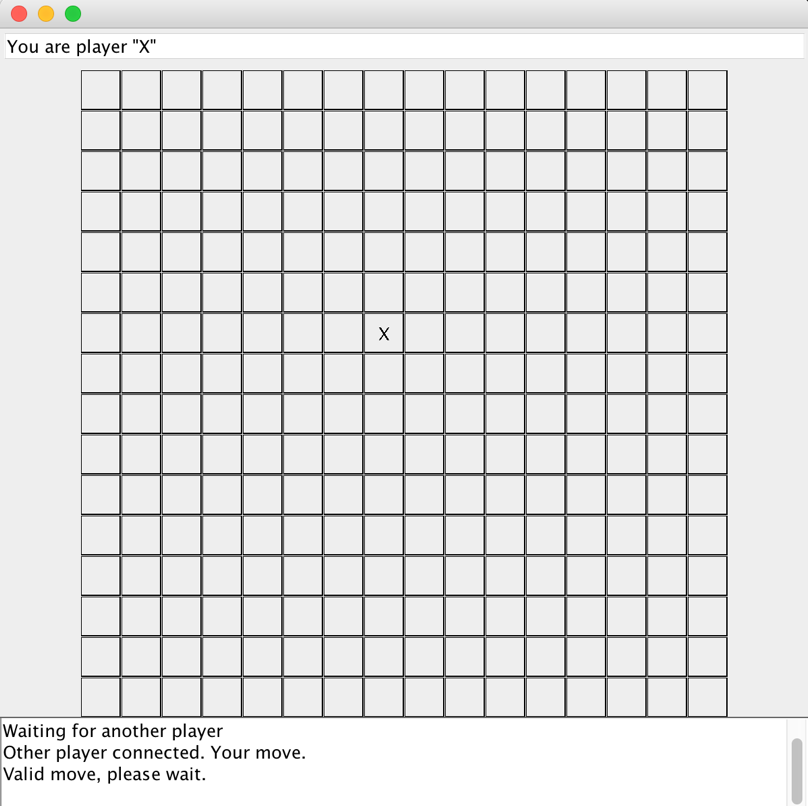
1. When run your program on a computer A, a GUI window pops up for a Client Player 1 who will place ‘X’ on the game board after each mouse click. The GUI window will NOT respond to any mouse click until the second Client Play 2 is connected and joined in the game.



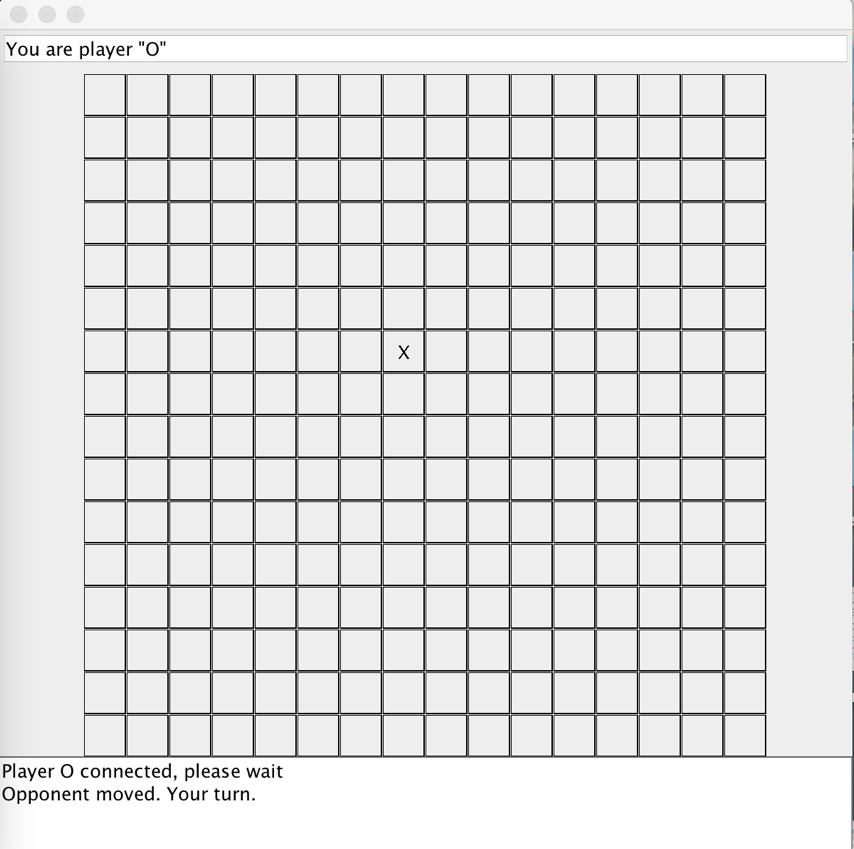
1. When run your program on a computer B, a GUI window pops up for a Client Player 2 who will place ‘O’ on the game board after each mouse click. Once the Player 2 is connected and joined in the game, the game is designed to always let Player 1 make the first move.



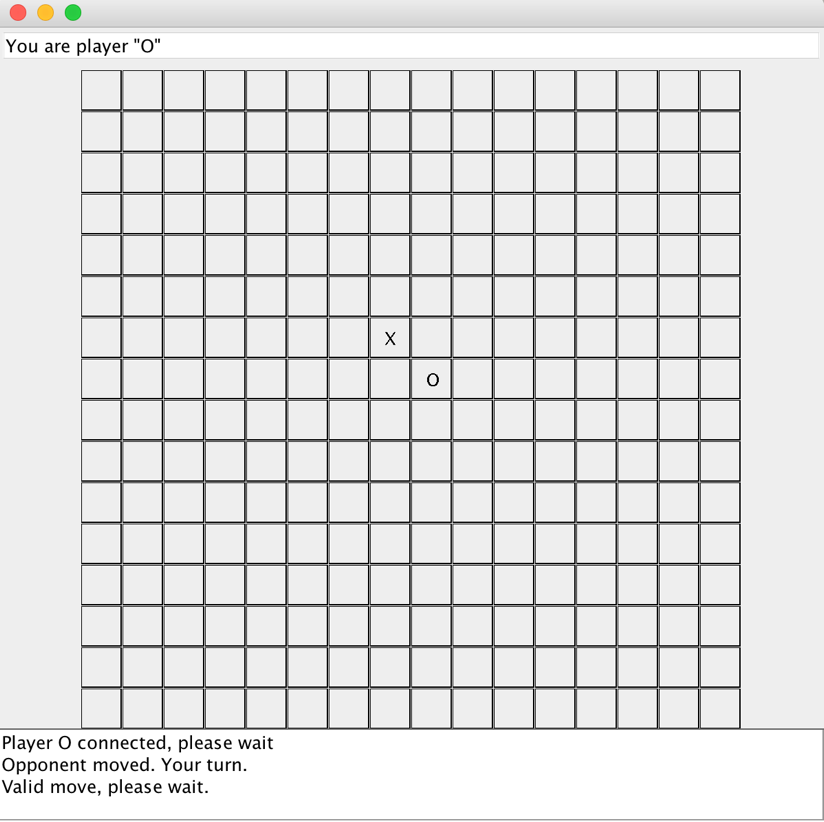
1. Then the Player 1 makes the first move by clicking one square on the game board, a symbol ‘X’ is placed on the square at the mouse click location, as shown in the picture below. After Player 1 has made the move, it has to wait until Player 2 finishes its move. During this wait, the GUI window of Player 1 will not respond to any mouse clicks.

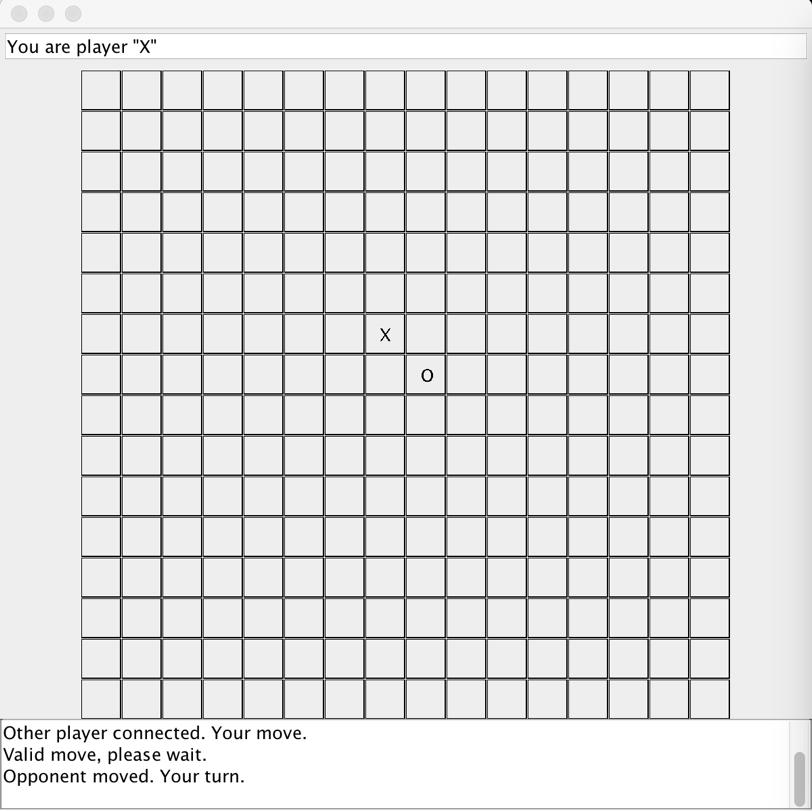


1. After Player 1 has made the move, as shown in the picture above, the clicked location and the symbol ‘X’ placed by the Player 1 have to be sent to Player 2 on Computer B for display in Player 2’s GUI window, which is shown in the picture below.

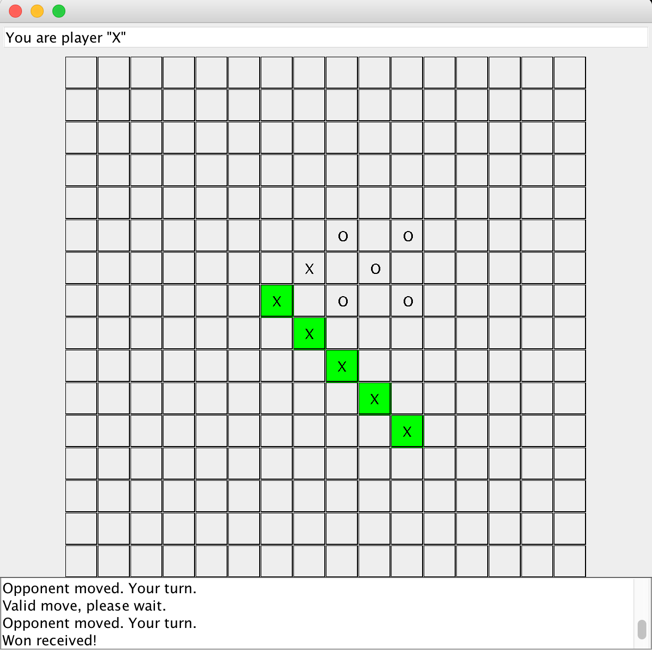


1. Then Player 2 will make the move, and the move has to be sent to Player 1 for display.

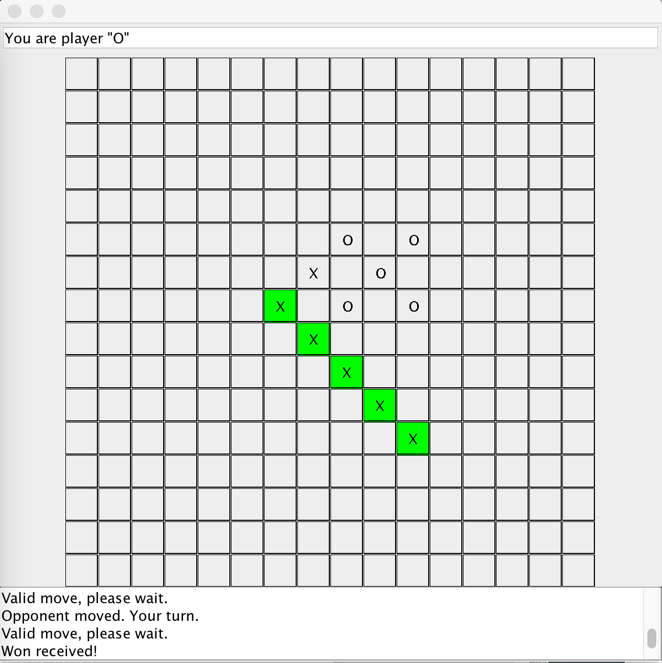




1. Then Player 1 and Player 2 alternate to make moves on the GUI window. The move made by the current player should always be displayed in the GUI window of the opponent, because two players play the same game and share the same game status on the board. During the current player waiting for the move of the opponent, the GUI window of the current player will not respond to any mouse clicks.
2. If the current player clicks on a previously occupied square on the game board, the current player will retain the privilege to make the next move (mouse click) until a valid location (vacant square) has been clicked.
3. The game continues until your client program detects a winning status. A winning status is reached when we find FIVE consecutive ‘X’ symbols or FIVE consecutive ‘O’ symbols on the game board in one of four directions, -- horizontally, vertically, diagonally, and inverse diagonally.



1. In the **example** above, the winning status is detected first by the player 1, then the five consecutive ‘X’ locations that mark the victory are highlighted by green background. Accordingly, these consecutive ‘X’ locations that mark the victory have to be sent to the. opponent player 2 for display



1. After a winning status is detected and the winning locations are highlighted in the GUI window of both players, the client program GUI for both players will not be responsive because the game is over.
2. The start-up code provided in the package is named as TicTacToeClient.java. You have the freedom to modify, to delete or to add any methods into that source file as needed. Also, you are allowed to create your own client program from the scratch as long as your program satisfies the requirements specified by the instructor.
3. You can have your own design for any details that have not specified in this document.