[CS209A-22Fall] Assignment 2 (100 points)

> Question Design: Yida Tao
>
> Code Sample: Xiang Yi
>
> Evaluation: Xiaofeng Wu
>
> Git & Code styles: Chi Xu

Deadline: 23:55pm Nov. 20

Late submissions after the deadline will NOT be accepted.

Tic-tac-toe Game

In this assignment, you'll be implementing a Tic-tac-toe game in client/server mode. In a Tic-tac-toe game, two players take turns marking the spaces in a three-by-three board with Xs or Os. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner[1]. For example, X is winner in the following game[1].

Implementation

We'll use **socket programming**, **multithreading**, and **JavaFX** to implement this game.

Server (35 points)

The server should wait for new players and automatically matches a newly connected player to another player to start a game. If there is no other player, the server should inform the newly connected player to wait.

The server should correctly track the status of each game and inform players when they win, lose or tie.

Client/Player (35 points)

A player should first connect to the game server and wait to be matched to another player. Upon successful matching, two players will start a new game by placing X and O in the 3x3 board alternately. Players should be informed when they win, lose, or tie.

GUI (15 points)

Primary GUI components of the game include the board, the Xs and Os. **Game GUI should be implemented using only JavaFX**. NO point will be given for this GUI part

- if you used only console for displaying the board
- if you used other GUI frameworks, such as Vue or Swing.

Exception Handling (15 points)

Many things could go wrong in a C/S mode game. A server may crash due to internal bugs; a player may quit intentionally or accidentally (e.g., due to network problem) without notifying the other player. You should handle such exception cases in your program, so that both clients and servers could react to unusual cases elegantly without affecting user experience.

Bonus (15 points)

- Account management (5 points): The server could support user registration and login. After login, users could check how many games s/he have played and how many times s/he has won.
- Opponent selection (5 points): After connected to the server, a player is able to see the list of players that are currently waiting to be matched. The player could choose the opponent from the list of players to start a new game.
- Resume game (5 points): After a client intentionally or accidentally quits in the middle of an ongoing game, he/she should be able to reconnect to the server and resume the game (from the status where the player quits) with the same opponent.

Reference

1. Tic-tac-toe: https://en.wikipedia.org/wiki/Tic-tac-toe

Demo

We provide a skeleton JavaFX code for you to get started. Please check here.

Submission

Please submit all of you `.java` files to the OJ system.

It doesn't matter how many `.java` files you have or which names you used for these files. However, please do NOT put all files into a zip. You should submit each `.java` one by one.

Evaluation

- **Functionalities**: You'll demonstrate your project during the lab session on Nov. 23, and we'll check whether you've accomplished the required functionalities onsite.
- **Version Control**: You should use `GitHub` to manage the code changes of your project (see lab 1 for further details of how to use `git`). You should made **at least 2 commits**. Your remote repo on GitHub **should be set to `private` before A2 deadline, so that no one else will see your code.**
- **Coding Style**: You should pay attention to write readable and maintainable code along the way. See lab 1 for how to use `CheckStyle` for that purpose. **After the deadline of A2**, you can set your GitHub repo to `public`, and we'll check whether any of your commits have reduced `CheckStyle` warnings according to `google_checks.xml`.