# Report

YUNTAO LU 1166487

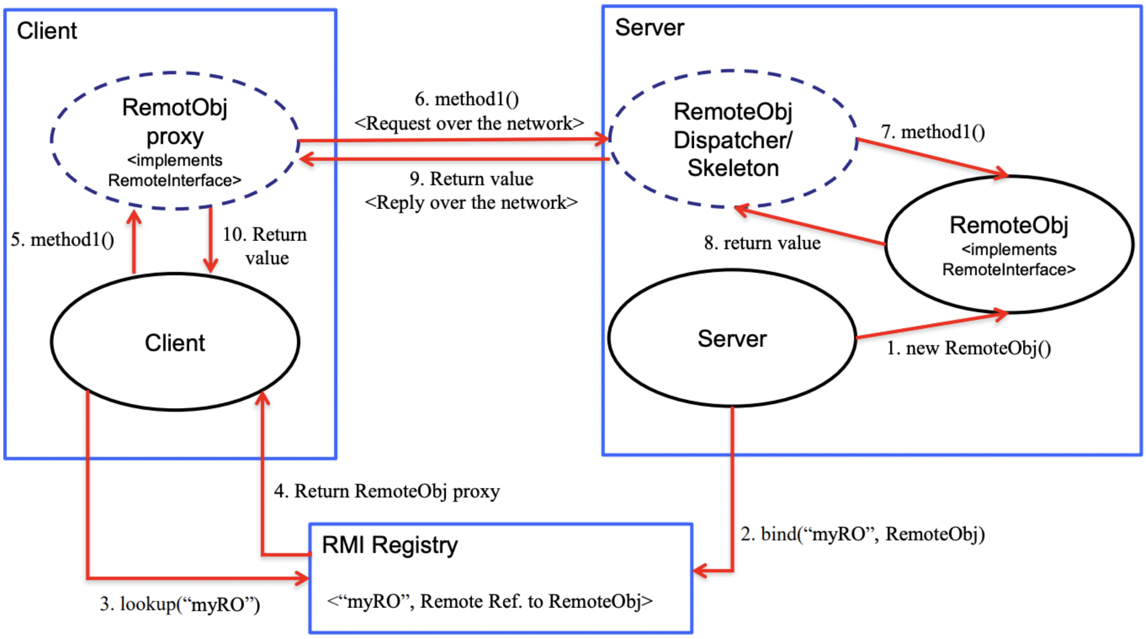
## Content

In this project we have to build a distributed Tic-Tac-Toe gaming system using Java, allowing multiple players to connect, play and chat with each other. There should be a GUI for the client and a server responsible for managing games. This system should be designed either using socket or Java RMI

## Components

### RMI

RMI is used to establish the connection between client and server. The server and client can interact with each other through the methods in the interface.

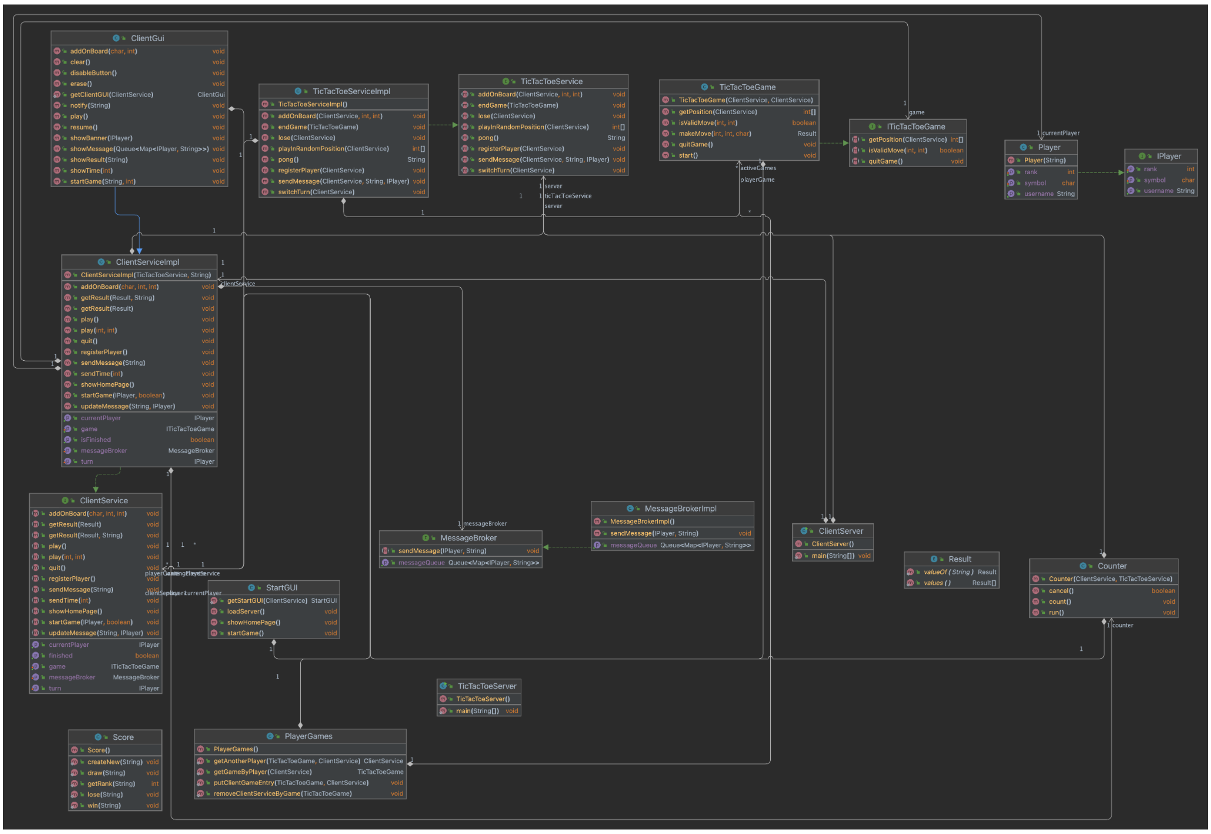

Client
RemotObj proxy <implements RemoteInterface>
5. method1()
10. Return value
Client
6. method1() <Request over the network>
9. Return value <Reply over the network>
Server
RemoteObj Dispatcher/ Skeleton
7. method1()
8. return value
RemoteObj <implements RemoteInterface>
Server
1. new RemoteObj()
4. Return RemoteObj proxy
2. bind("myRO", RemoteObj)
RMI Registry
<"myRO", Remote Ref. to RemoteObj>
3. lookup("myRO")

### Concurrency

Multithreading and concurrency are used to guarantee the parallel processing of the application

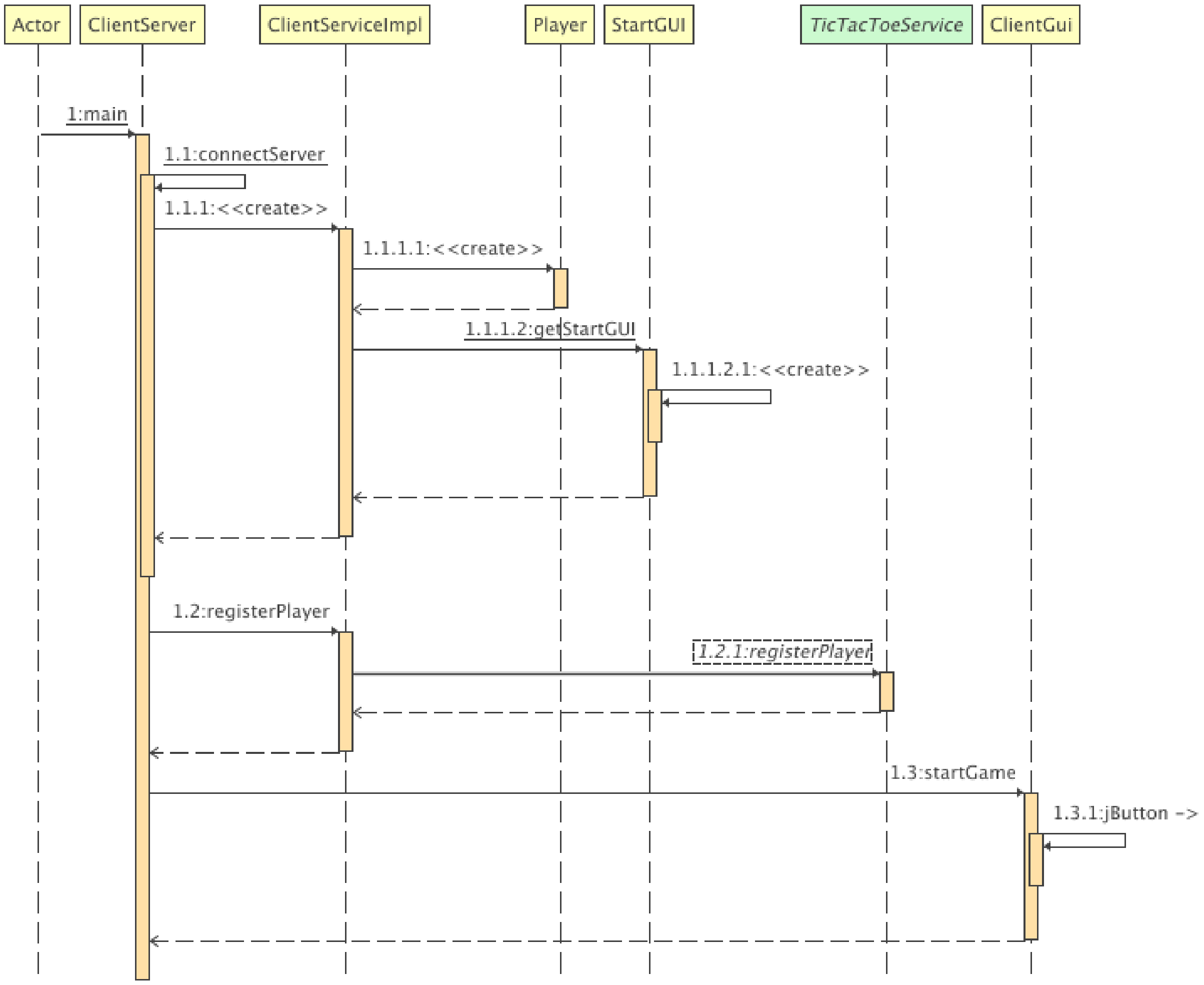
# Class Design

## Class Diagram

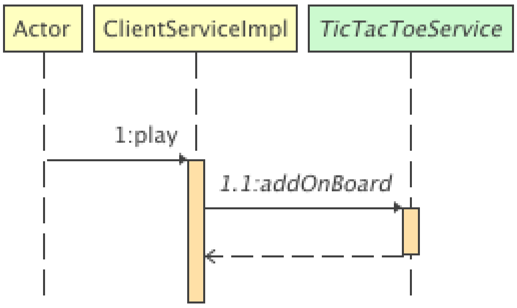

ClientGui
addOnBoard(char, int) void
clear() void
disableButton() void
erase() void
getClientGUI(ClientService) ClientGui
notify(String) void
play() void
resume() void
showBanner(IPlayer) void
showMessage(Queue<Map<IPlayer, String>>) void
showResult(String) void
showTime(int) void
startGame(String, int) void
TicTacToeServiceImpl
TicTacToeServiceImpl()
addOnBoard(ClientService, int, int) void
endGame(TicTacToeGame) void
lose(ClientService) void
playInRandomPosition(ClientService) int[]
pong() String
registerPlayer(ClientService) void
sendMessage(ClientService, String, IPlayer) void
switchTurn(ClientService) void
TicTacToeService
addOnBoard(ClientService, int, int) void
endGame(TicTacToeGame) void
lose(ClientService) void
playInRandomPosition(ClientService) int[]
pong() String
registerPlayer(ClientService) void
sendMessage(ClientService, String, IPlayer) void
switchTurn(ClientService) void
TicTacToeGame
TicTacToeGame(ClientService, ClientService)
getPosition(ClientService) int[]
isValidMove(int, int) boolean
makeMove(int, int, char) Result
quitGame() void
start() void
ITicTacToeGame
getPosition(ClientService) int[]
isValidMove(int, int) boolean
quitGame() void
Player
Player(String)
rank int
symbol char
username String
IPlayer
rank int
symbol char
username String
game
currentPlayer
server
ticTacToeService
activeGames
playerGame
clientService
ClientServiceImpl
ClientServiceImpl(TicTacToeService, String)
addOnBoard(char, int, int) void
getResult(Result, String) void
getResult(Result) void
play() void
play(int, int) void
quit() void
registerPlayer() void
sendMessage(String) void
sendTime(int) void
showHomePage() void
startGame(IPlayer, boolean) void
updateMessage(String, IPlayer) void
currentPlayer IPlayer
game ITicTacToeGame
isFinished boolean
messageBroker MessageBroker
turn IPlayer
ClientService
addOnBoard(char, int, int) void
getResult(Result) void
getResult(Result, String) void
play() void
play(int, int) void
quit() void
registerPlayer() void
sendMessage(String) void
sendTime(int) void
showHomePage() void
startGame(IPlayer, boolean) void
updateMessage(String, IPlayer) void
currentPlayer IPlayer
finished boolean
game ITicTacToeGame
messageBroker MessageBroker
turn IPlayer
StartGUI
getStartGUI(ClientService) StartGUI
loadServer() void
showHomePage() void
startGame() void
MessageBroker
sendMessage(IPlayer, String) void
messageQueue Queue<Map<IPlayer, String>>
messageBroker
MessageBrokerImpl
MessageBrokerImpl()
sendMessage(IPlayer, String) void
messageQueue Queue<Map<IPlayer, String>>
ClientServer
ClientServer()
main(String[]) void
Result
valueOf ( String ) Result
values ( ) Result[]
Counter
Counter(ClientService, TicTacToeService)
cancel() boolean
count() void
run() void
counter
TicTacToeServer
TicTacToeServer()
main(String[]) void
Score
Score()
createNew(String) void
draw(String) void
getRank(String) int
lose(String) void
win(String) void
PlayerGames
PlayerGames()
getAnotherPlayer(TicTacToeGame, ClientService) ClientService
getGameByPlayer(ClientService) TicTacToeGame
putClientGameEntry(TicTacToeGame, ClientService) void
removeClientServiceByGame(TicTacToeGame) void

# Client

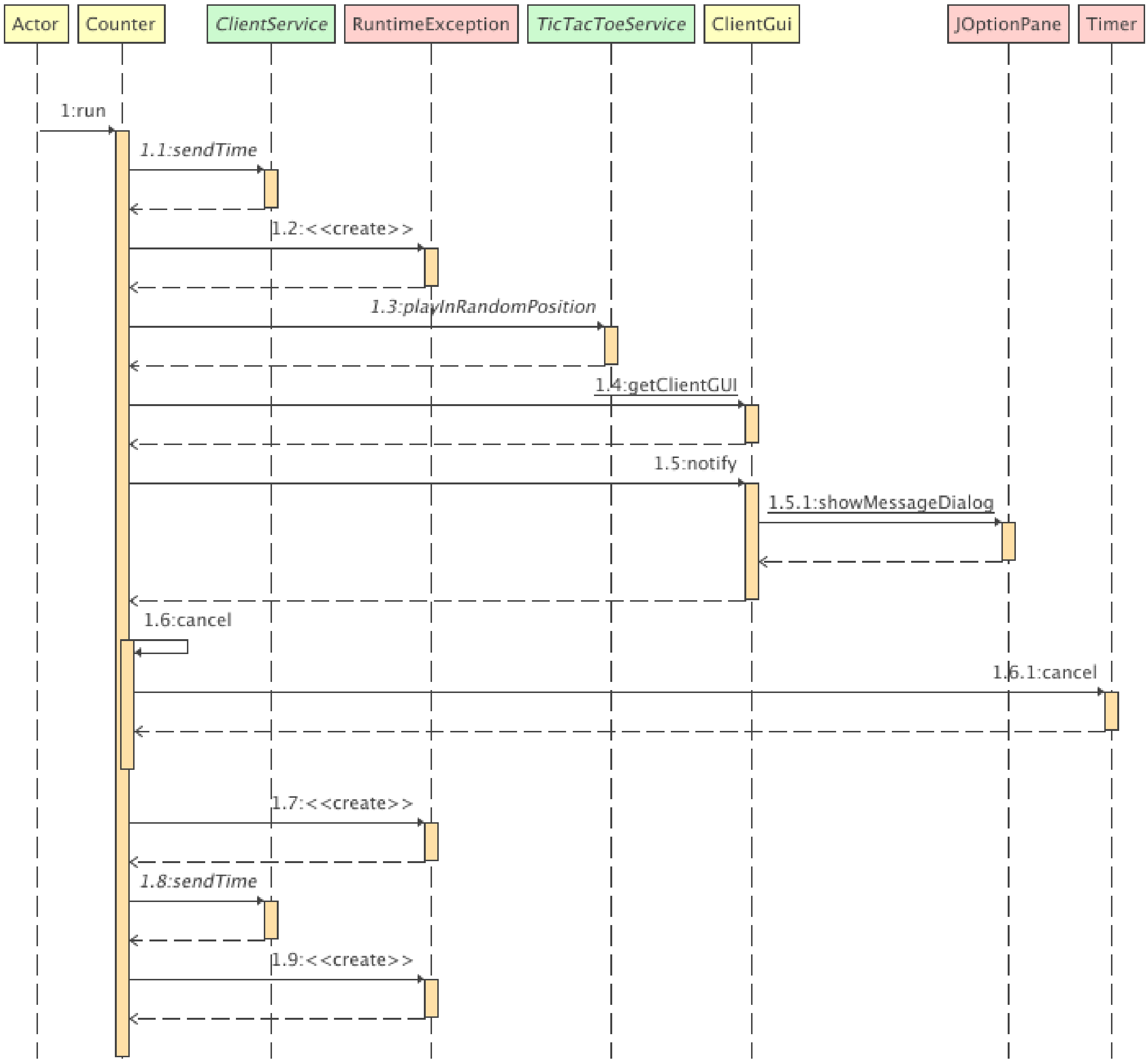
## Connect to the server



## Play game

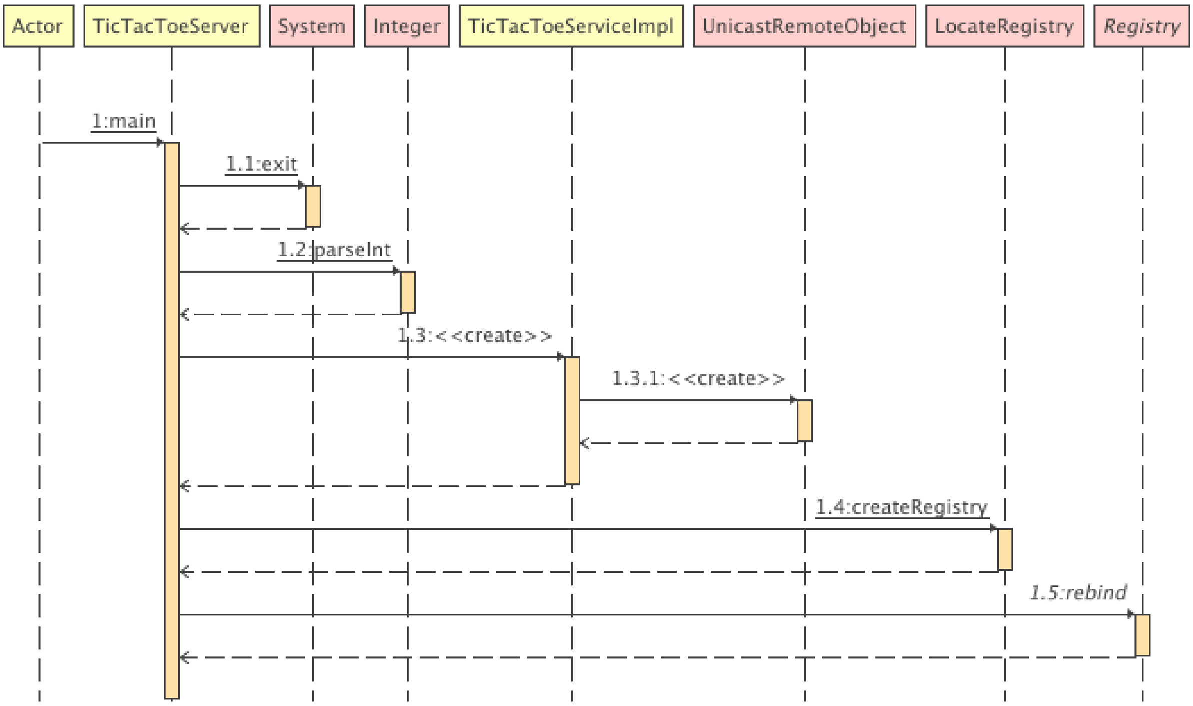


### Countdown

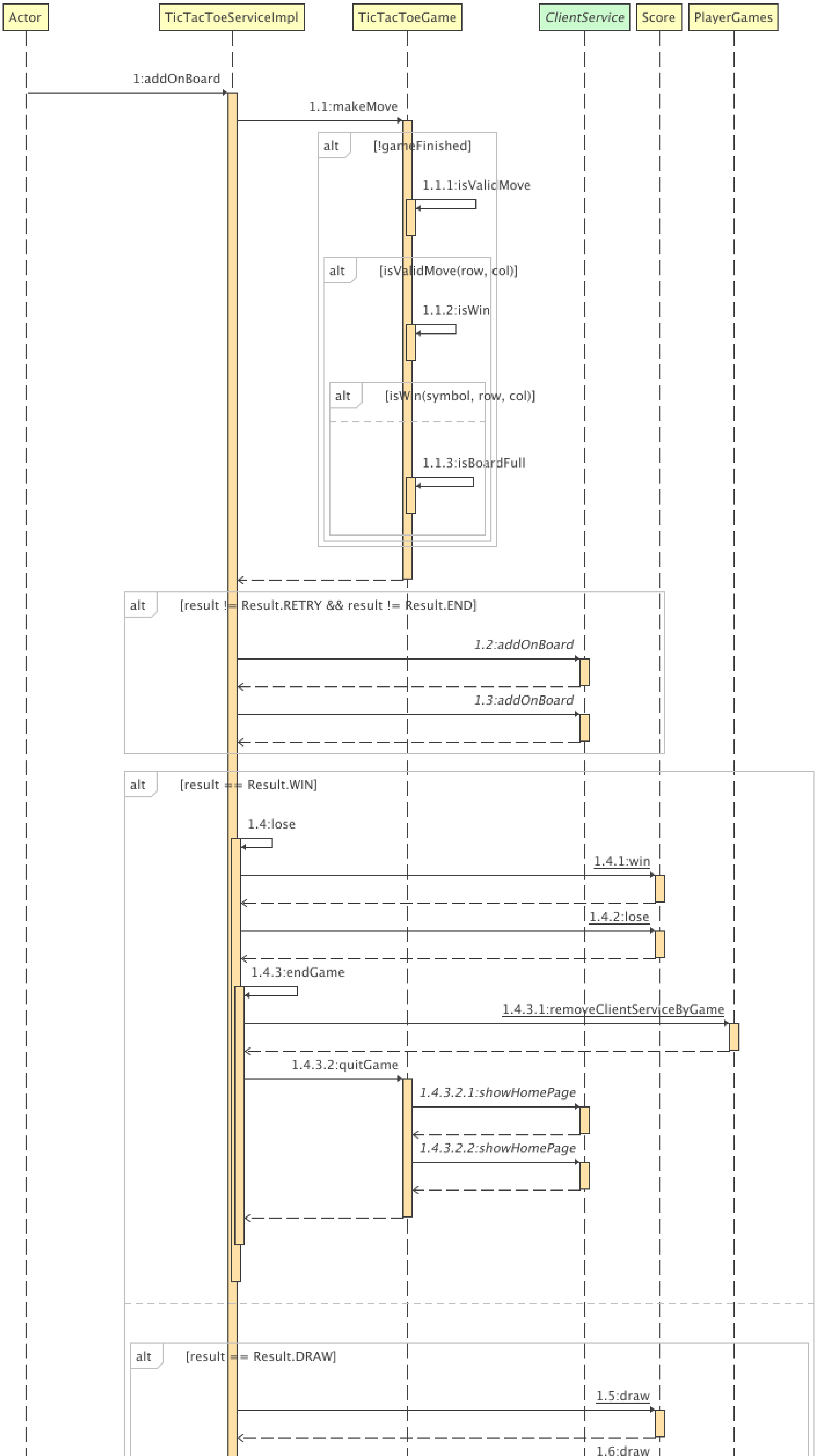


## Server

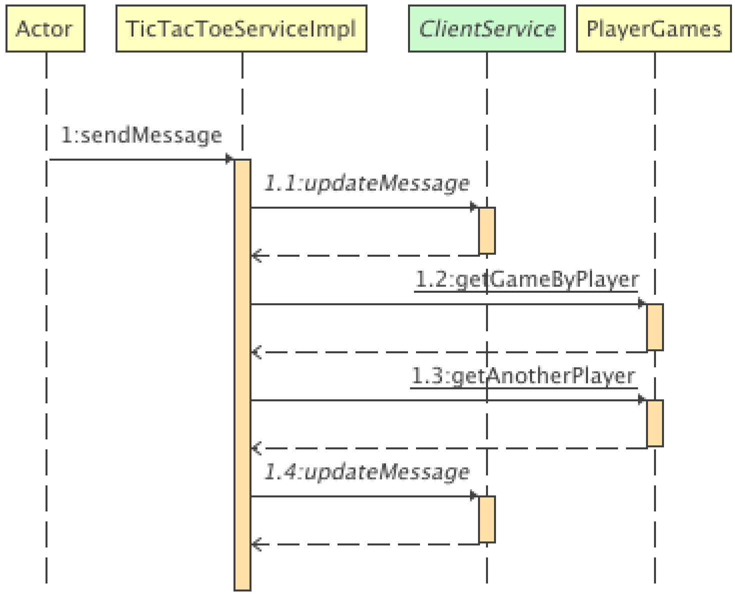
### Set up server


Actor
TicTacToeServer
System
Integer
TicTacToeServiceImpl
UnicastRemoteObject
LocateRegistry
Registry
1:main
1.1:exit
1.2:parseInt
1.3:<<create>>
1.3.1:<<create>>
1.4:createRegistry
1.5:rebind

Play game

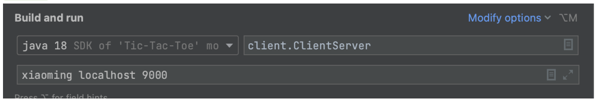
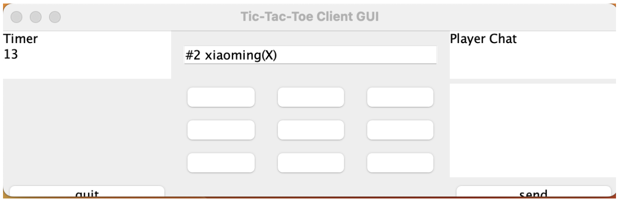
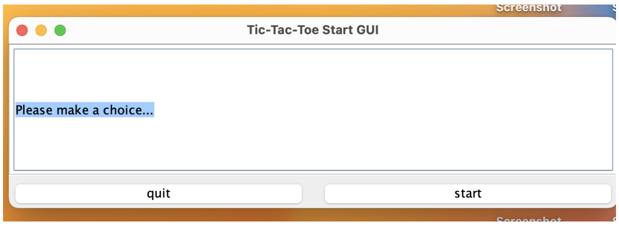

Actor
TicTacToeServiceImpl
TicTacToeGame
ClientService
Score
PlayerGames
1:addOnBoard
1.1:makeMove
alt
[!gameFinished]
1.1.1:isValidMove
alt
[isValidMove(row, col)]
1.1.2:isWin
alt
[isWin(symbol, row, col)]
1.1.3:isBoardFull
alt
[result != Result.RETRY && result != Result.END]
1.2:addOnBoard
1.3:addOnBoard
alt
[result == Result.WIN]
1.4:lose
1.4.1:win
1.4.2:lose
1.4.3:endGame
1.4.3.1:removeClientServiceByGame
1.4.3.2:quitGame
1.4.3.2.1:showHomePage
1.4.3.2.2:showHomePage
alt
[result == Result.DRAW]
1.5:draw
1.6:draw

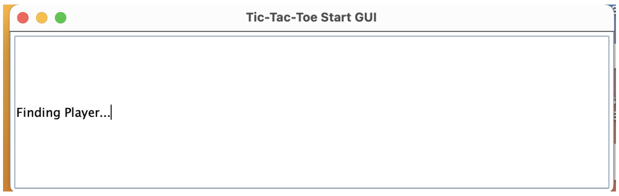
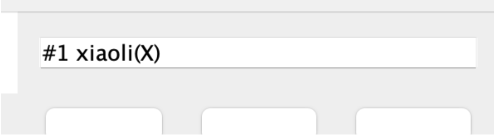
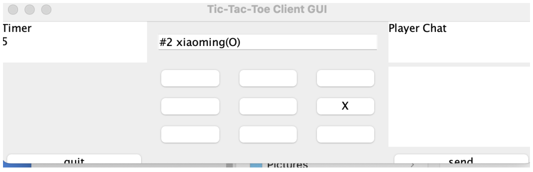
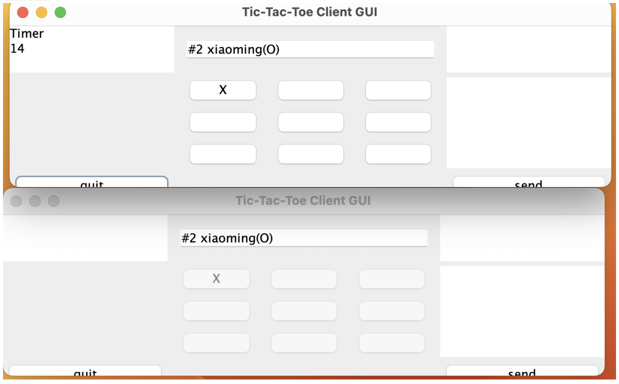
### Sending message

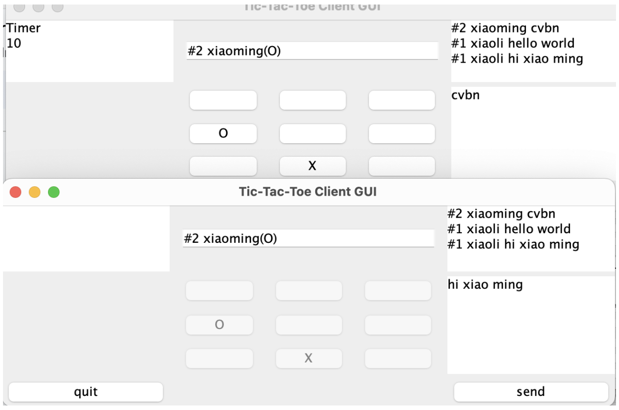


# Analysis

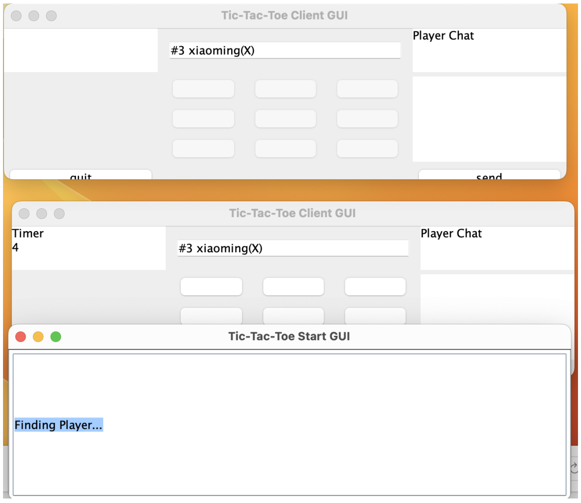
## Client

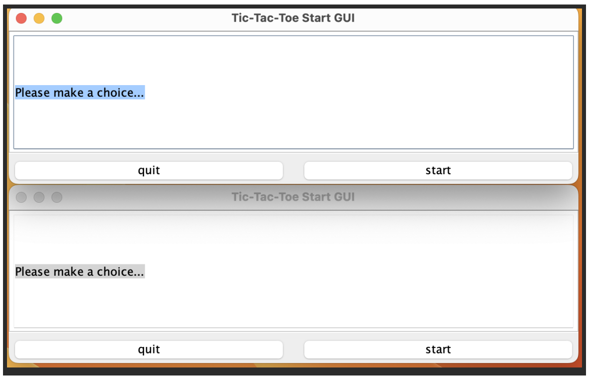
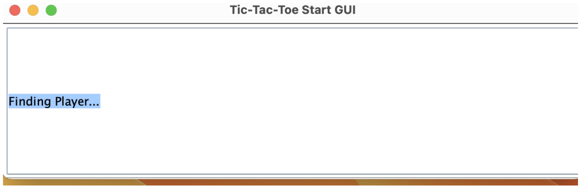
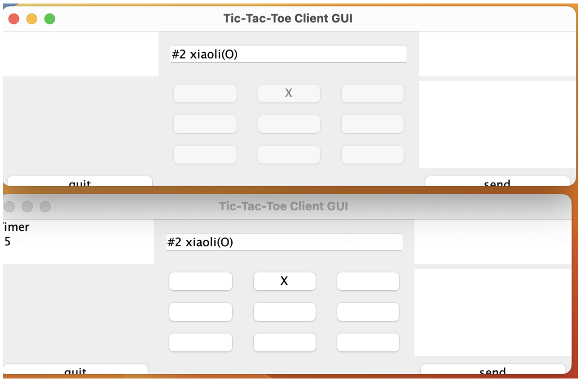
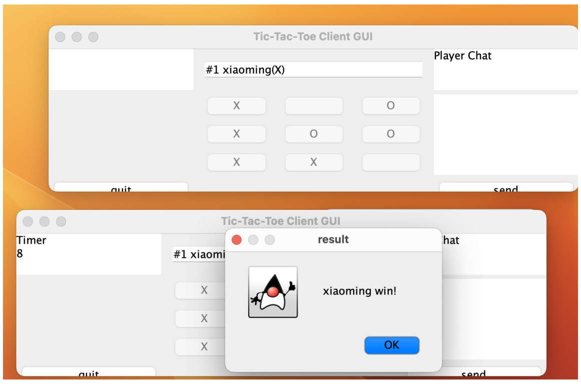
| | Requirement | Result |
|---|---|---|
| 1 | When the client application is launched for the first time, a player should be able to pass Their username as a command-line argument. | ✅<br><br> <br>java 18 SDK of 'Tic-Tac-Toe' mo<br>client.ClientServer<br>xiaoming localhost 9000 |
| 2 | A simple graphical interface (GUI) that visually represents the Tic-Tac-Toe game board in the middle,including the current state of the game | ✅Start GUI and Client GUI are created to indicate the state of the game according to the figure<br><br> Tic-Tac-Toe Client GUI<br>Timer<br>Player Chat<br>quit<br>send<br>13<br>#2 xiaoming(X)<br><br> <br>Please make a choice... |

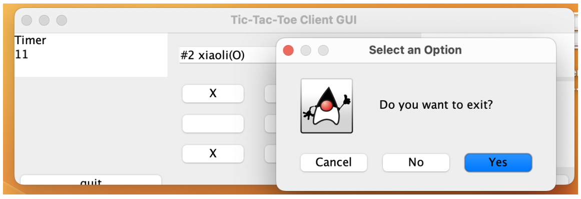
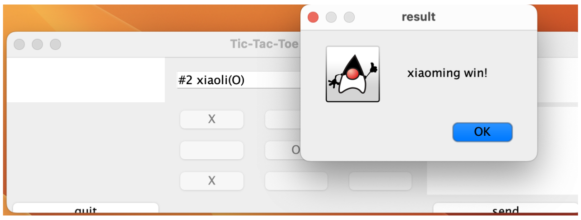
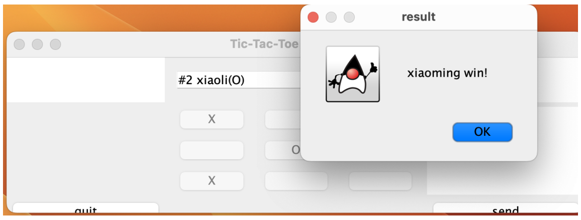
| | | |
|---|---|---|
| 3 | As soon as the client launches the program, the player should see an empty board, with a text on top of the board showing "Finding Player" before the server finds a match. | ✅The massage “Finding Player”  is shown when the server finds the match<br><br> <br>Finding Player... |
| 4 | When a player has a turn, they may click on an empty cell of the Tic-Tac-Toe board their assigned symbol should be drawn to that cell to indicate their move. | ✅A 3*3 tables of buttons are given to allow players to click on and the symbol assigned is drawn on the cell. |
| 5 | After a game starts, a text should be shown on top of the board showing the current player’s name (the player who has the turn to make a move), with an ‘X’  or ‘O’ written beside their name to indicate the symbol that has been assigned to them by the server. | ✅<br><br>#1 xiaoli(X) |
| 6 | The client should then inform the current move to the server, and then the turn will be switched to the other player | ✅<br><br> Tic-Tac-Toe Client GUI<br>Timer<br>5<br>#2 xiaoming(O)<br>Player Chat<br>X<br>quit<br>Pictures<br>send |
| 7 | When two players start a new game, they will take turns making moves on the Tic-Tac-Toe board, Which should be visually reflected on both of their client GUIs. | ✅<br><br> Tic-Tac-Toe Client GUI<br>Timer<br>14<br>#2 xiaoming(O)<br>X<br>quit<br>send<br>Tic-Tac-Toe Client GUI<br>#2 xiaoming(O)<br>X<br>quit<br>send |
| 8 | There should be a chat window (on the right side of the GUI) that enables players to communicate with their opponents in real time. Only the last 10 messages should be displayed in the chat window, where the most recent message should appear at the bottom. | ✅ |

| | | |
|---|---|---|
| | | <br>Tic-Tac-Toe Client GUI<br>Timer<br>10<br>#2 xiaoming(O)<br>#2 xiaoming cvbn<br>#1 xiaoli hello world<br>#1 xiaoli hi xiao ming<br>cvbn<br>Tic-Tac-Toe Client GUI<br>#2 xiaoming(O)<br>#2 xiaoming cvbn<br>#1 xiaoli hello world<br>#1 xiaoli hi xiao ming<br>hi xiao ming<br> |

## Server

| | Requirement | Result |
|---|---|---|
| 1 | The server should be able to handle multiple clients (players) that join the system and want to play the game | ✅ |
| 2 | When a new client (player) joins the system, they will be added to a pool of active players in the system. | ✅ |
| 3 | From the pool of active players, the server should match any two players to start a new game. The Server should continue to match players whenever there are active players in the pool. For an odd number of players, the last remaining player should wait until another player joins the system (the server may communicate this information to the corresponding client so that it keeps showing "Finding Player" ). | ✅The players registered should be added to the active players pool. The server should match the players randomly. If there are odd number of players, the last one should wait till the new players enter<br><br>Tic-Tac-Toe Client GUI<br>#3 xiaoming(X)<br>Player Chat<br>Tic-Tac-Toe Client GUI<br>Timer<br>4<br>#3 xiaoming(X)<br>Player Chat<br>Tic-Tac-Toe Start GUI<br>Finding Player...<br> |
| 4 | When a game starts, the server may randomly select which player should take the first turn, and assign a<br>symbol to each player (either ‘X’ or ‘O’) | ✅ |
| 5 | After a game is finished, if a player agrees to find a new game, they should be moved to the pool of active players again | ✅When a game is finished, it will go back to the home page |

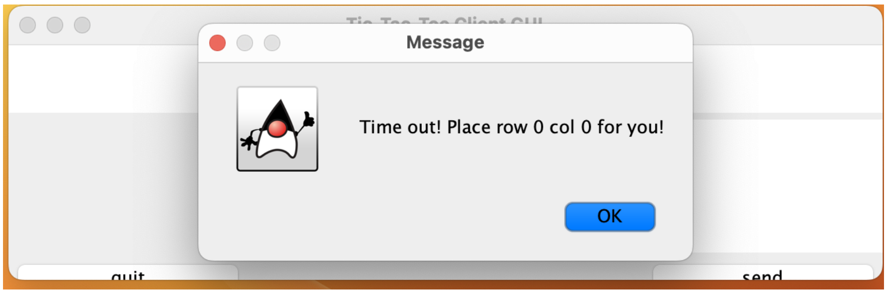
| | | |
|---|---|---|
| | | <br> Tic-Tac-Toe Start GUI<br>quit<br>start<br>Tic-Tac-Toe Start GUI<br>quit<br>start<br>Please make a choice...<br>Please make a choice...<br>The player will be added to the active player pool again when the player click the start button<br><br> <br>Finding Player... |
| 6 | he server should continuously update the game's state as the players make moves, and communicate The game's current state to the players after each move | ✅The server is continuously checking the status of the game, if a player make a move, it will disable the buttons and take turns to the other player<br><br> Tic-Tac-Toe Client GUI<br>#2 xiaoli(O)<br>X<br>quit<br>send<br>Tic-Tac-Toe Client GUI<br>imer<br>5<br>#2 xiaoli(O)<br>X<br>quit<br>send |
| 7 | The server should also check for a winning condition after each move. When the game is finished (a player wins, or the game ends as a draw), the server should communicate the result to both players. | ✅The server continuously check the winning state, and it will inform the both players if one of them wins, or there is a draw<br><br> Tic-Tac-Toe Client GUI<br>Player Chat<br>#1 xiaoming(X)<br>X<br>O<br>X<br>O<br>O<br>X<br>X<br>quit<br>send<br>Tic-Tac-Toe Client GUI<br>Timer<br>8<br>#1 xiaomi<br>X<br>X<br>X<br>hat<br>result<br>OK<br>quit<br>send<br>xiaoming win! |

| | | |
|---|---|---|
| 8 | If a player quits the system in the middle of a game, their current match will be forfeited and the other player will be declared as the winner. | ✅If one player click the quit button, there would be a option dialog confirming whether to quit. If it clicks yes, the other party will win this game and they will go back to the home page. If it clicks no, the play will continue<br><br> Tic-Tac-Toe Client GUI<br>Timer<br>11<br>#2 xiaoli(O)<br>X<br>X<br>quit<br>Select an Option<br>Do you want to exit?<br> <br><br> Tic-Tac-Toe<br>#2 xiaoli(O)<br>X<br>O<br>X<br>quit<br>send<br>result<br>xiaoming win!<br> <br><br> Tic-Tac-Toe<br>#2 xiaoli(O)<br>X<br>O<br>X<br>quit<br>send<br>result<br>xiaoming win!<br> |

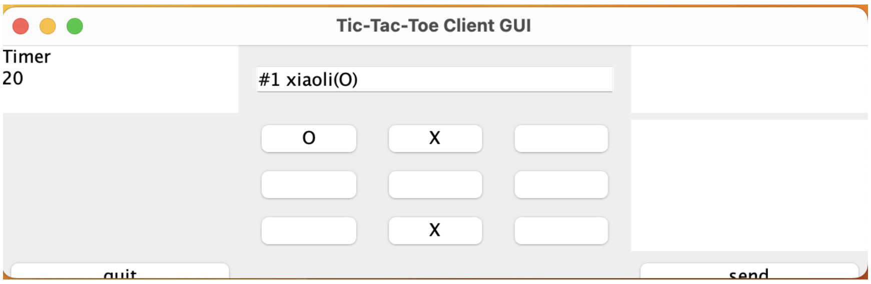
## Additional feature

### Timing ✅

Each player should have at most 20 seconds to complete a move during the turn. If the time runs out and the player has not made a move, it should randomly choose any empty cell of the board, draw a player symbol and inform the move to the server. The turn will be switched to the next player



Time out! Place row 0 col 0 for you!



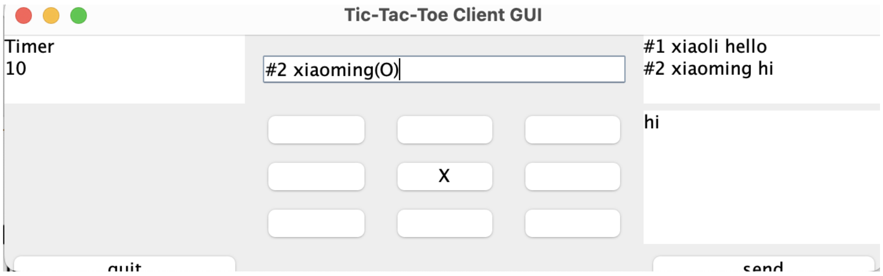
Tic-Tac-Toe Client GUI

Timer
20

#1 xiaoli(O)

## Ranking ✅

- Each player should be assigned a rank # based on the total rating points they accumulated.
- The player starts with point 0 and get 5 for each win, minus 5 for each lose and +2 for each draw.
- The rank should be visible in front of their username and the chat messages



Tic-Tac-Toe Client GUI

Timer
10

#2 xiaoming(O)

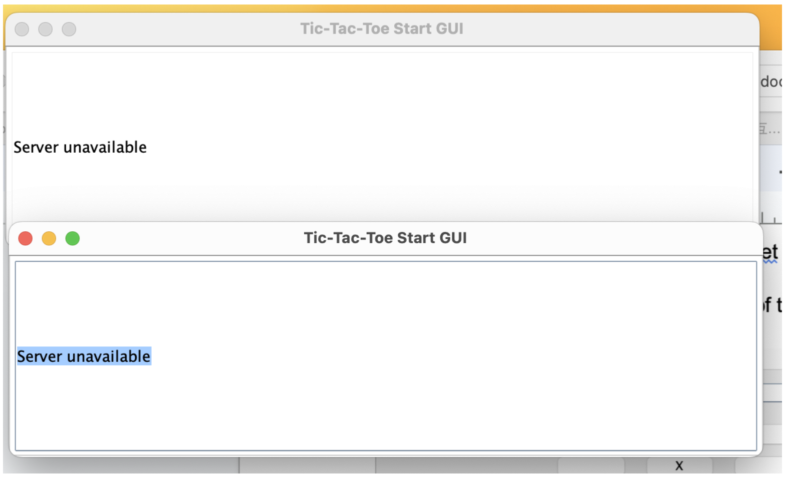
#1 xiaoli hello
#2 xiaoming hi

hi

## Fault Tolerant

### Server Side✅

During the server crash all the players should see the message "Server unavailable" and then the clients will exit gracefully after 5 seconds.



Tic-Tac-Toe Start GUI

Server unavailable

Tic-Tac-Toe Start GUI

Server unavailable

### Client Side⭕

**Requirement**: During the client crash, the game will be paused for 30 seconds. If the client rejoins the system during the client, the game will resume. Otherwise the game will be ended as a draw

**Analysis**
I have not implemented this function very well. The main reason for it is that I have not created a data structure to track the username and its related clientService at the early stage, and check whether it’s a new user or the existing user who wants to reconnect. There is no suitable object to store the status of the game as well. More work needs to be done in this area.

## Conclusion

Both server and client applications are designed and implemented under the principle of low coupling and high cohesion. Each class is designed to focus on the specific tasks. Concurrency and multithreading are applied to the application in order to allow the application to process multiple tasks at the same time. In order to improve the application design further, a database may need to be added to store the information of each player. More works should be done to track the status of client.