# CHE NEWSLETTERDB form | Qatar University

**Qatar University**

**College of Engineering**

**Computer Science and Engineering**

**Operating System**

**CMPS405 - Spring 2024**

**Section: LB52**

**Date: 5May 2024**

**Student Name: Noor Ahmed QUID: 201905802**

**Student Name: Thoraya Aldosari QUID: 201703993**

**Student Name: Maha Albazeli QUID: 201901699**

# 

**Contributions**:

1. Server à Thoraya
2. Client à Maha
3. Game à Noor

**Challenges**

Challenge 1 :

Comprehending Socket Programming: At first, it was difficult to understand the notion of socket programming. not in sending and receiving data, as we are skilled in doing so in the lab, but rather with the utility of the data itself.  
Challenge 2 :

Swing-Based GUI Design: Creating the graphical user interface (GUI) required additional work. In order to make the display appear neat and user-friendly, we had to find out how to organize the components, manage user input, and update the display constantly.  
Challenge 3:

Multiplayer Environment Synchronization: It was difficult to oversee several clients connecting with the server at once. It took precise planning and testing to ensure appropriate synchronization in order to prevent race situations or data corruption.

Challenge 4:

Error Handling and Exception Management: It might be intimidating to deal with errors and exceptions, particularly when it comes to network connectivity. Reliable error-handling procedures had to be put in place in order to deal with unanticipated events like lost connections or incorrect input.

Challenge 5:

Testing and Debugging Across systems: Unexpected problems were found when the program was tested on several networks and systems. It was our responsibility to troubleshoot compatibility problems and guarantee that the code operated consistently regardless of the operating system or network setup.

Issues :

Issue 1 :

Initially, managing several client connections at once caused the user interface to become ineffective. The UI experienced lag and became unresponsive as a result. The Solution: Threading was introduced to manage client-server communication asynchronously in order to overcome this problem. Even when numerous clients were connected, the user interface stayed responsive because each client connection was handled in its own thread.

**public** **void** run() {

**try** {

**while** (**true**) {

Socket socket = serverSocket.accept();

executor.submit(**new** Player(socket));

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

Issue 2 :

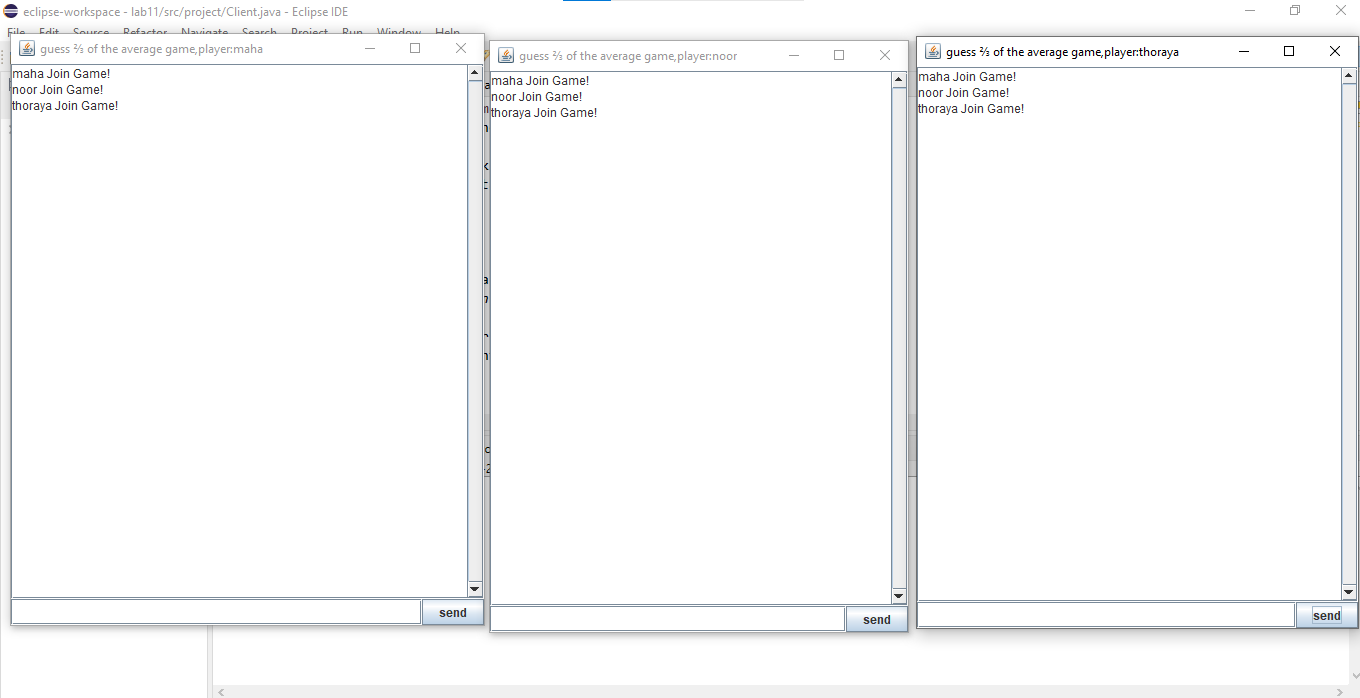
Initially, there were compatibility problems with the application when it ran in various contexts or operating systems, which led to inconsistent functioning or behaviour.what. we did is To ensure cross-platform compatibility, platform-specific configurations and dependencies were found and fixed. This required testing the application across a range of platforms and adjusting as needed to account for variations in system setups.

Issue 3 :

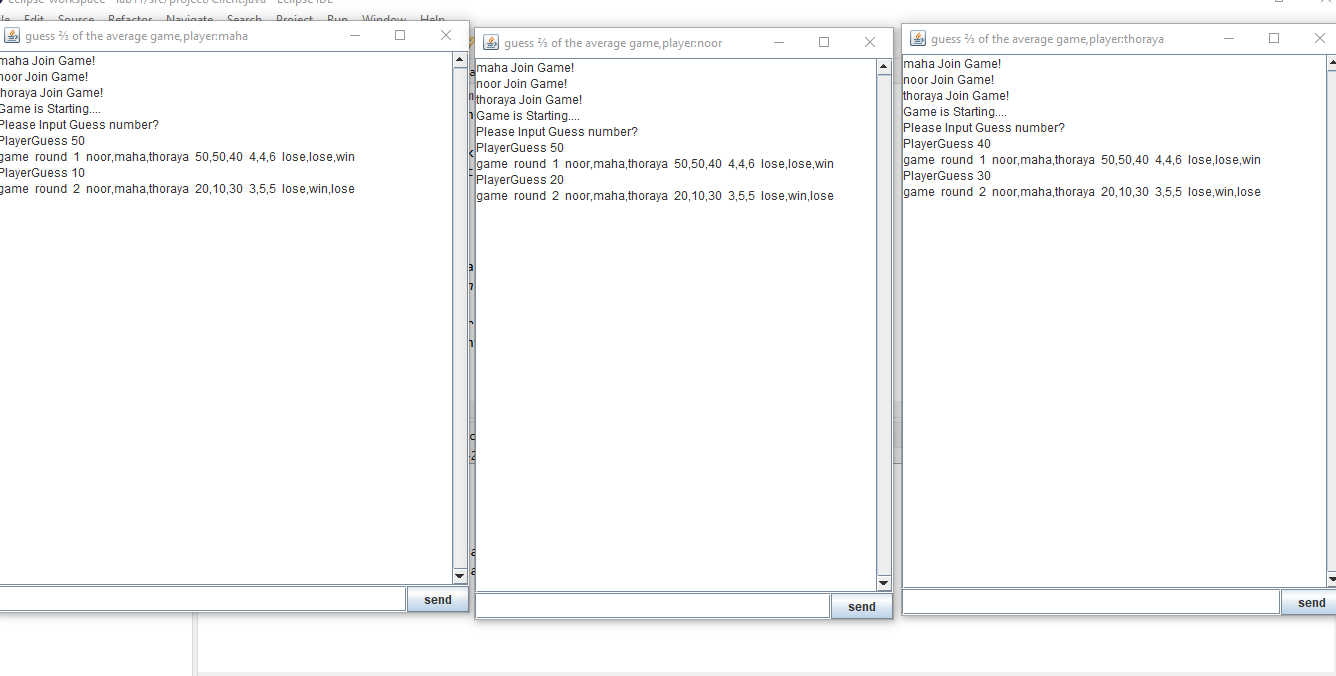
Problem: Ineffective Player Removal: During the development of a multiplayer game, I encountered ineffective player removal when a player lost all of their points. There could have been performance problems as the number of participants rose because the procedure for locating and eliminating these players was not optimized.

I Optimize the Player Removal Process as a solution.

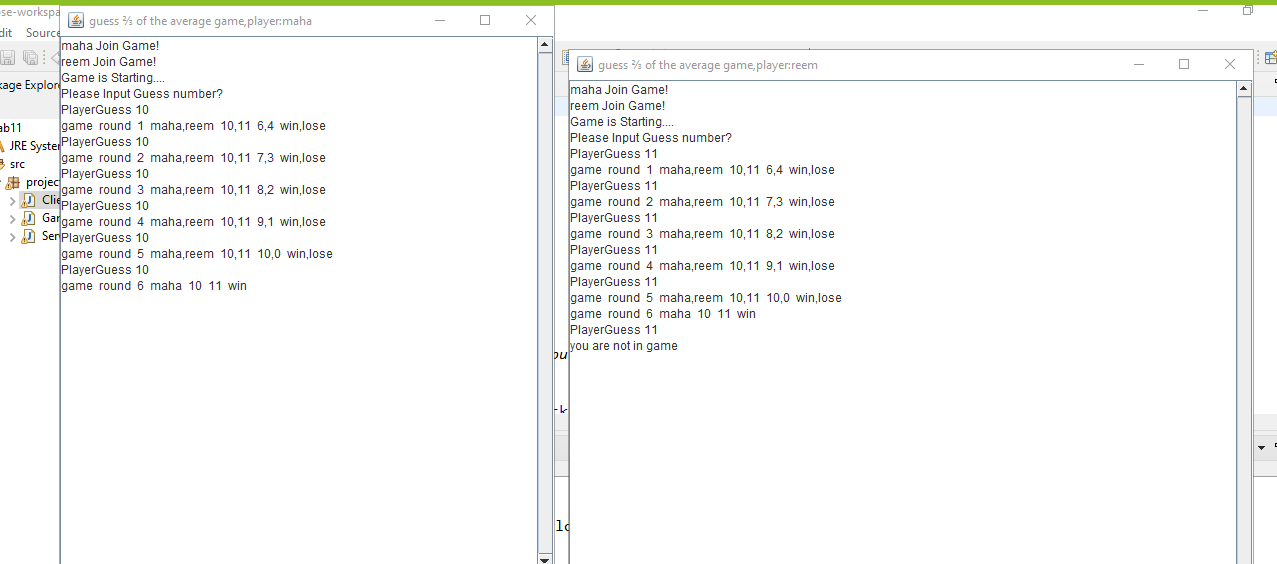
**Screenshots of game:**

****

**To start the game the players should enter “Join” to enter the game.**



**After the all players join one of them should enter “Start” then each player should guess the number. For example Maha write “PlayerGuess 50”**



**If the player loses all points and reach 0, he will be exit from the game.**

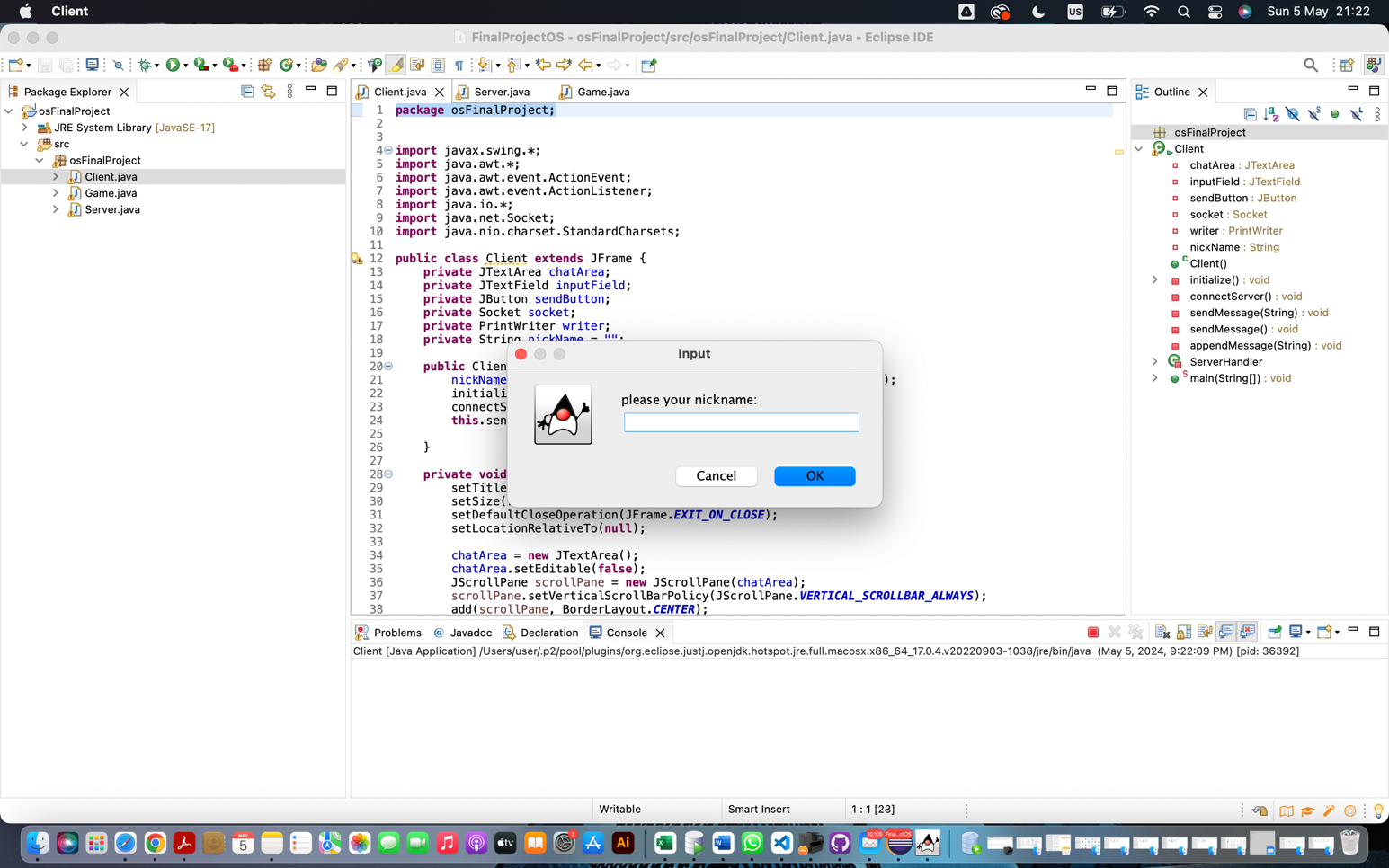


**The player can chat with each other any time by enter “SendMeesage” then the player’s name that will reactive the message then message.**

**Another Screenshots from different device:**

**Noor’s device**

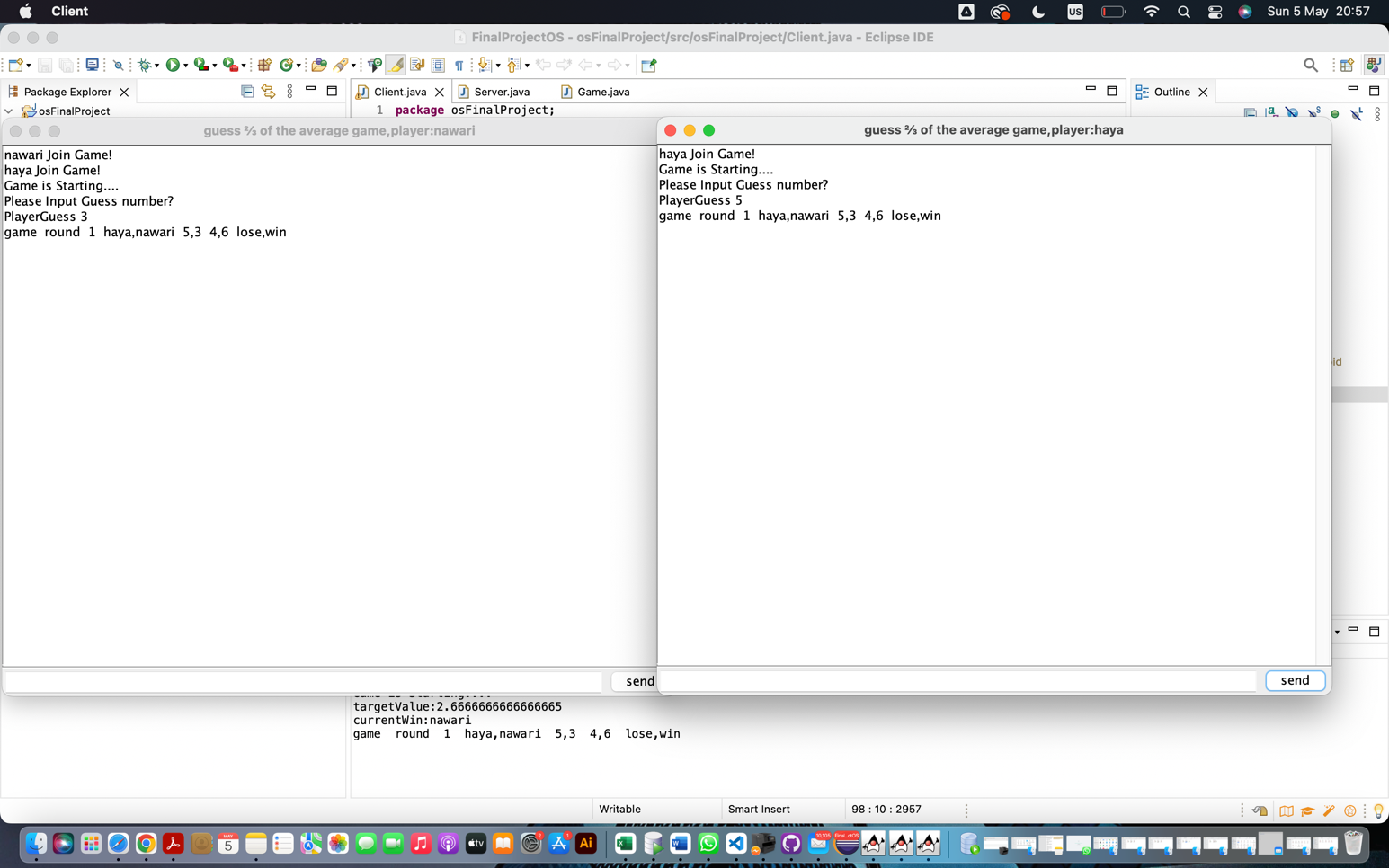
**Firstly I run the code and entered players names**

****

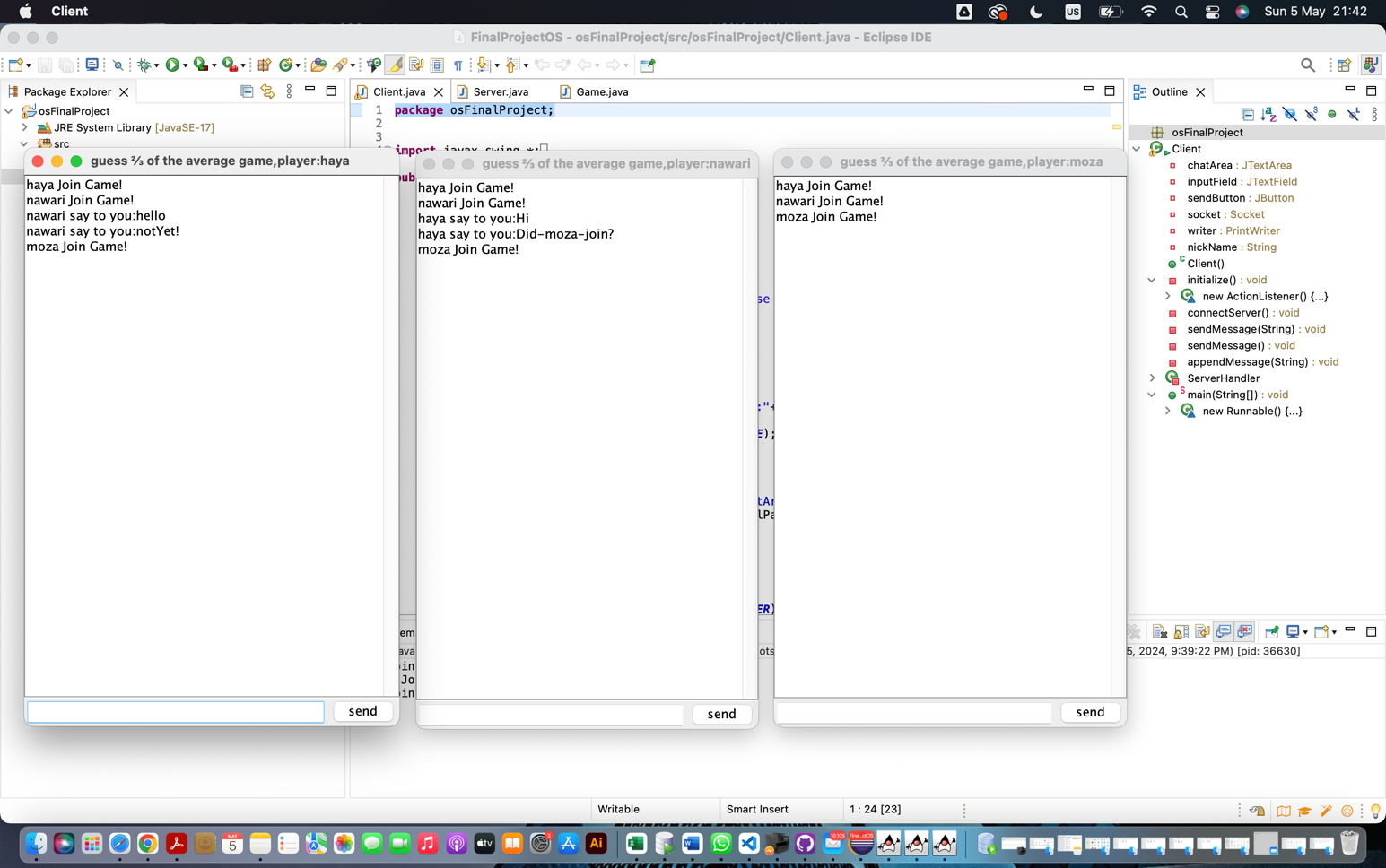
**After that I let the players join the game by typing Join**

**One of them should enter Start for the game to start**

**Then the players can start the game by guessing the numbers**

****

More than two players can join and Players can communicate using SendMessage with the player name to communicate with each other which makes the game more fun and challenging



if one player joined and wants to start the game a message will be displayed asking for more people to join

