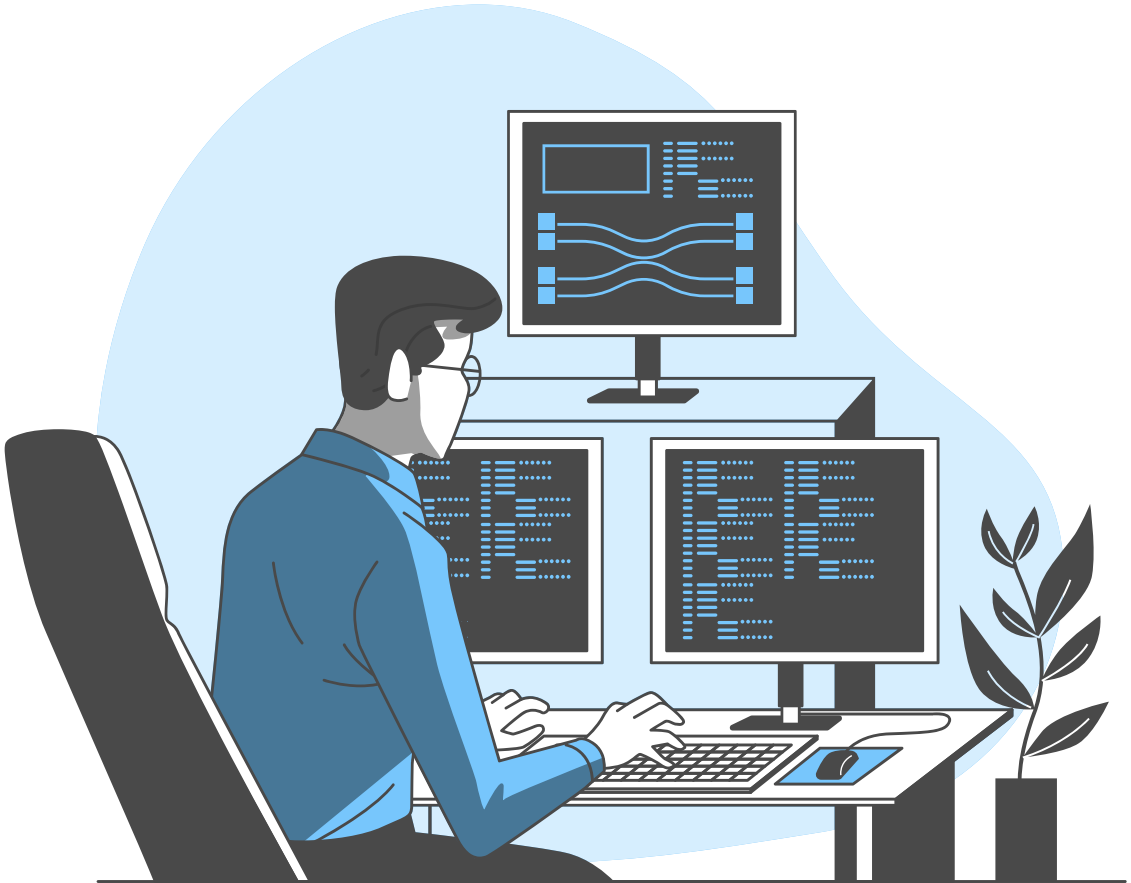


# Computer Networks Project

## Connect Four Game through Multi-client implementation

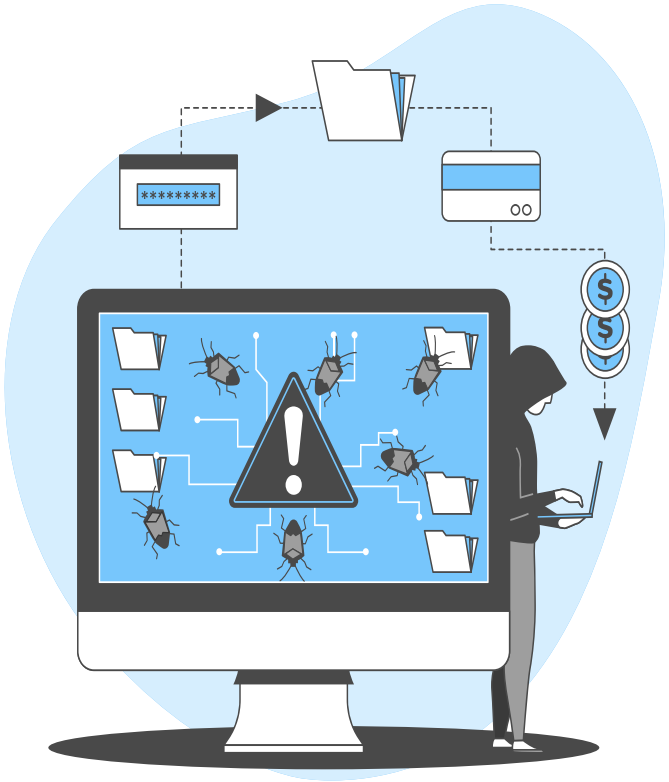
# Process Definition

In our project we have implemented a multiplayer game (Connect Four).

For our multiplayer game, we will follow the multiclient-server game architecture.

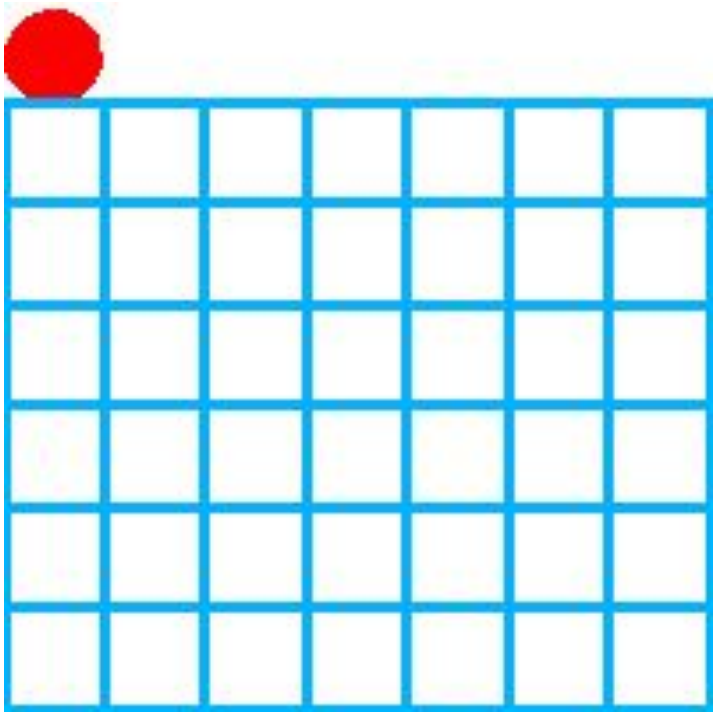
Here, the client is responsible for displaying the game to the player, handling the player's input, and for communicating with the server.

The server, on the other hand, is responsible for broadcasting that data to each client and maintaining integrity. We have used threads to manage multiple clients.



# What is Connect Four?

- Connect-Four is a tic-tac-toe-like two-player game in which players alternately place pieces on a vertical board with columns and rows (8 x 6 for our project).
- Each player uses pieces of a particular color (red and blue for our project), and the object is to be the first to obtain four pieces in a horizontal, vertical, or diagonal line.
- Because the board is vertical, pieces inserted in a given column always drop to the lowest unoccupied row of that column.
- As soon as a column contains 6 pieces, it is full and no other piece can be placed in the column.



# Project Journey

01

## Socket Programming

We have done socket programming in Java using threads

02

## Building GUI

Designed an interactive GUI interface for the game

03

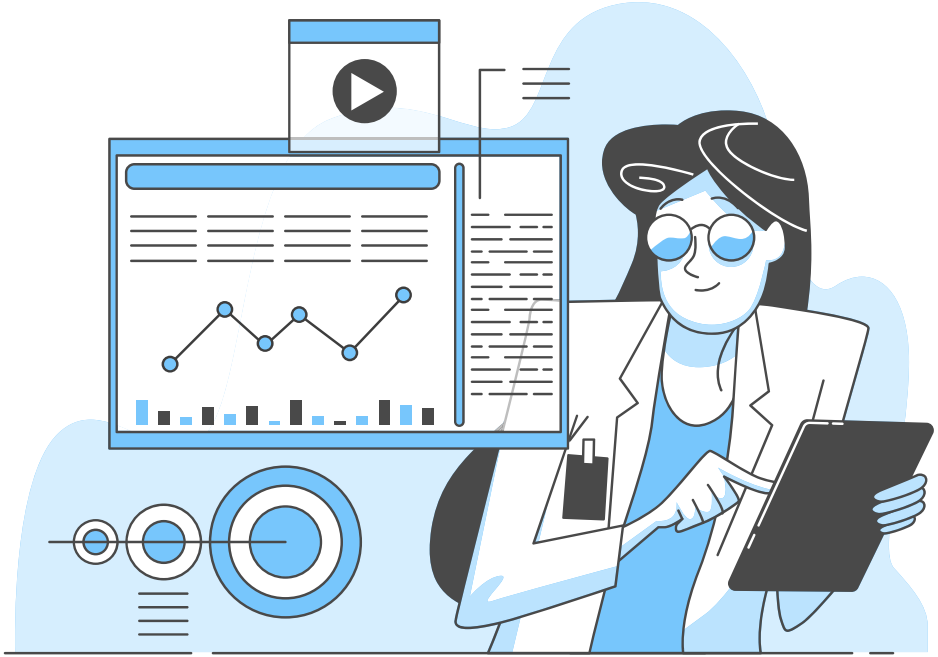
## Error Handling

Keep a check on all the possible moves of the users

04

## Tried to make project scalable

Server can hold multiple game plays



# What is socket programming?

Socket programming is a way of connecting two or more nodes on a network to communicate with each other. One socket (node) listens on a particular port at an IP, while other sockets reach out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

# Why to use threads?

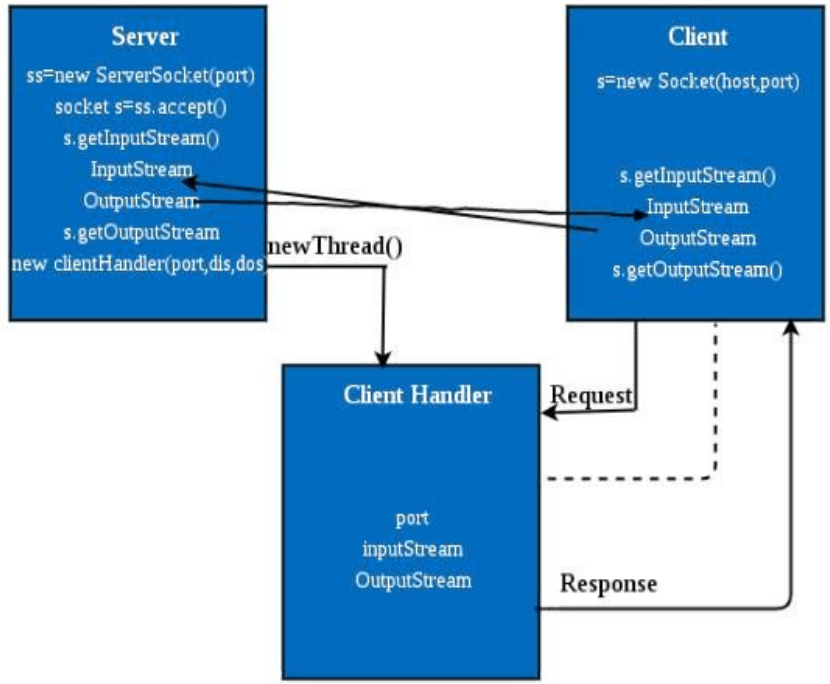
The reason is simple, we don’t want only a single client to connect to server at a particular time but many clients simultaneously. We want our architecture to support multiple clients at the same time. For this reason, we must use threads on server side so that whenever a client request comes, a separate thread can be assigned for handling each request.

# Steps of implementation

1. **Establishing basic connection between the two clients through the server.**

   Server class:

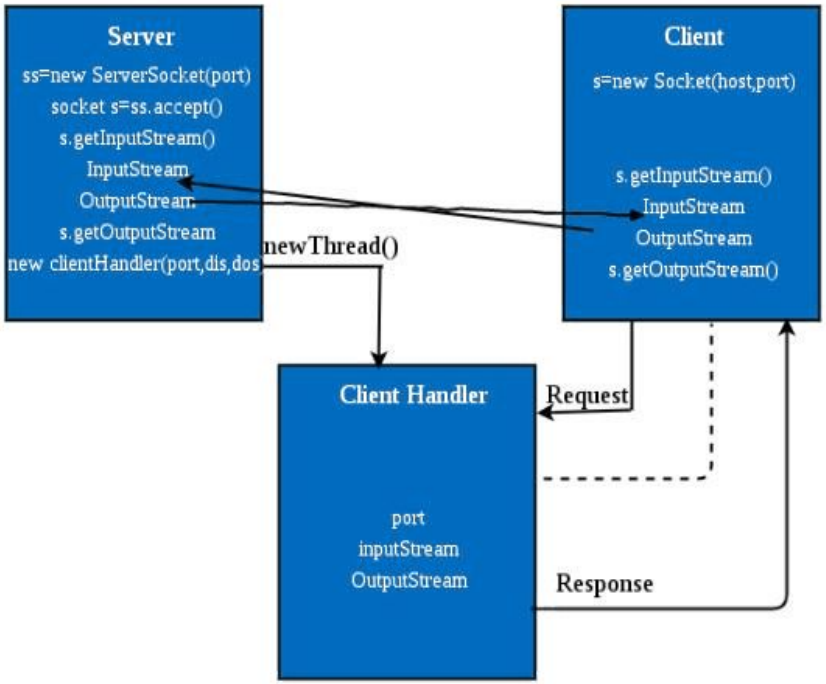
   - Establishing the Connection
   - Obtaining the Streams
   - Creating a handler object
   - Invoking the start() method



# Steps of implementation

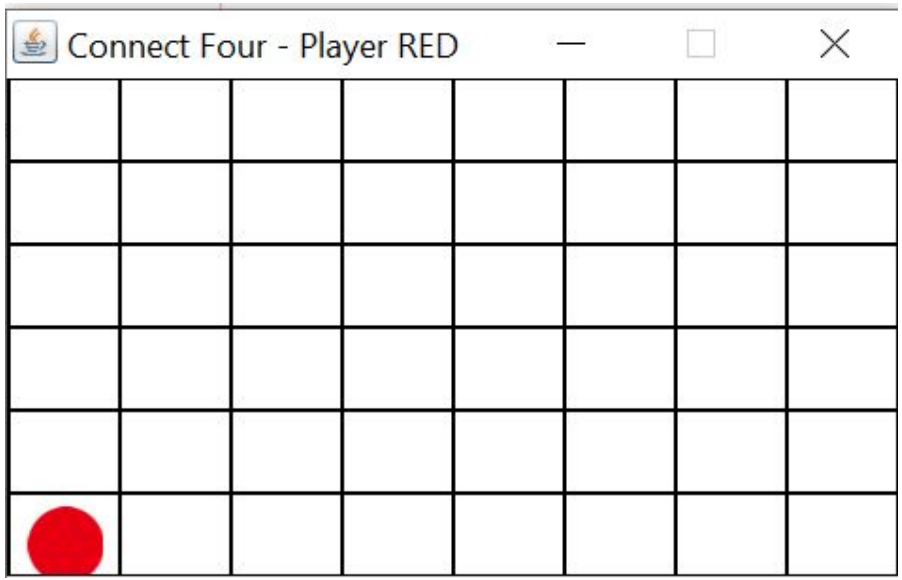
ClientHandler class : An object of this class will be instantiated each time a request comes.

- Extending Thread class
- Making the constructor of this class with unique Socket, DataInputStream and DataOutputStream parameters
- Writing the run() method of this class



# Steps of implementation

2. Build basic GUI
3. Make it interactive by adding mouse-listeners
4. Add necessary icon png resources and assign the colours to the GUI
5. Implement the vertical strategy in a single client using an algorithm



# Steps of implementation

**6. Create a basic text based protocol for communication between the clients through the server and vice-versa.**

**7. Implement the protocol so that the client can wait for their turn and make their move.**

**8. Implement the protocol so that every move made by client A is also reflected for client B.**

We followed the CFP (Connect Four protocol) that is more or less text-based.
The messages that are sent are as follows.:

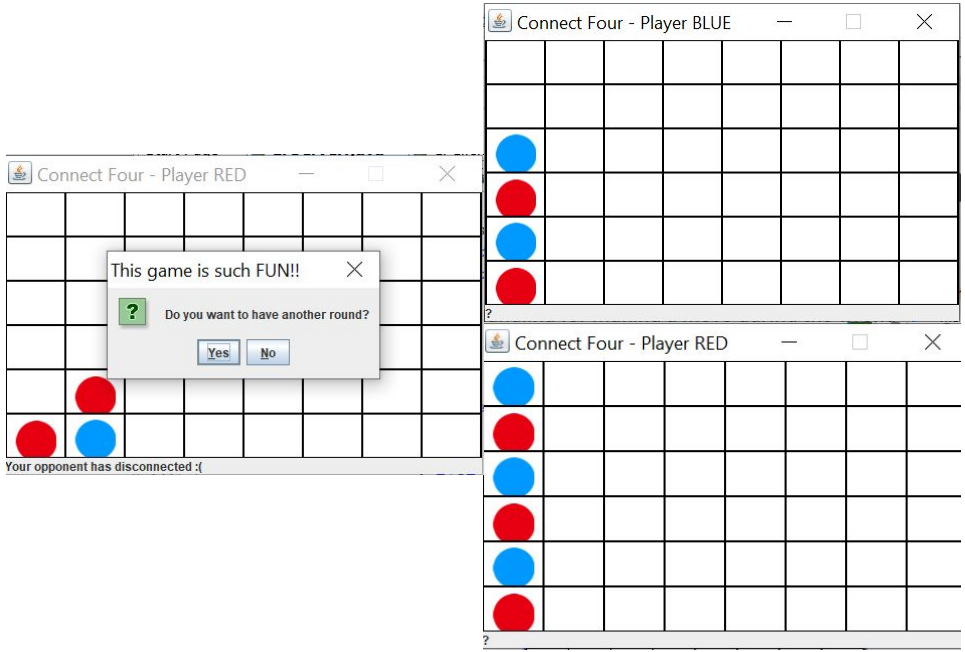
| Client -> Server | Server -> Client |
| --- | --- |
| MOVE <n> (0 <= n <= 48) | HELLO <String> (String in {"RED", "BLUE"}) |
| QUIT | MOVE_CORRECT |
| | RIVAL_MOVED <n> |
| | WIN |
| | LOSS |
| | TIE |
| | MSG <text> |

# Steps of implementation

**9. Error handling for making a move during the wrong turn.**

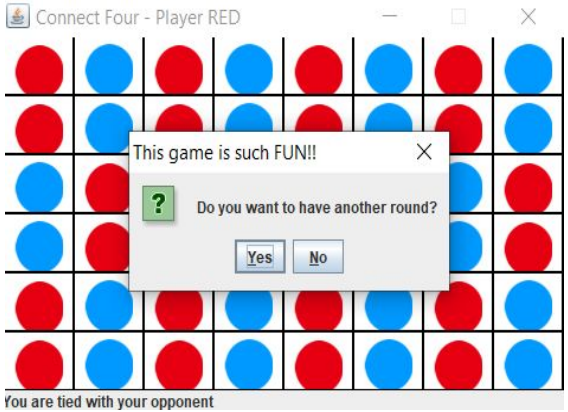
**10. Error handling for inserting a checker in a row that is full.**

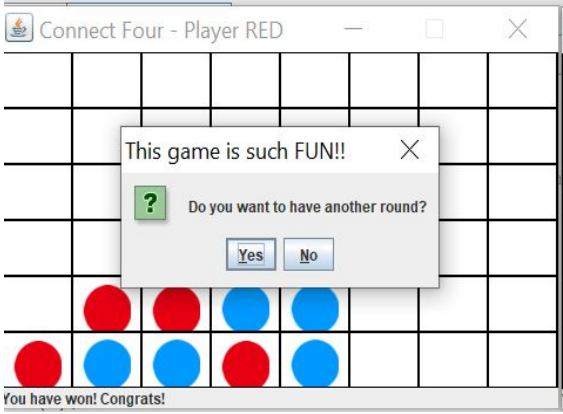
**11. Error handling for when one of two players disconnect.**

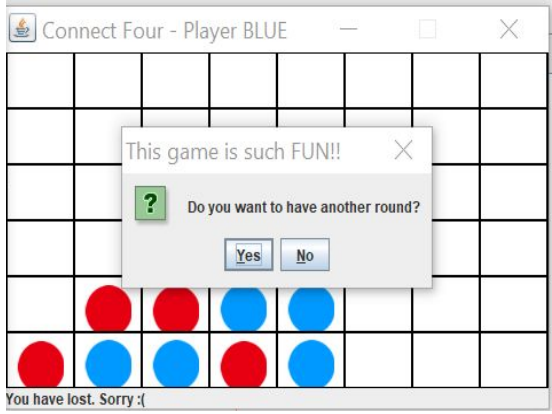


# Steps of implementation

11. Employ an algorithm to constantly check if any client has made a successful row of four checkers.
12. Employ an algorithm to constantly check if the matrix is full.
13. If 7 or 8 method return true value, the game will be over.
14. Give the player a chance to play again if they wish.



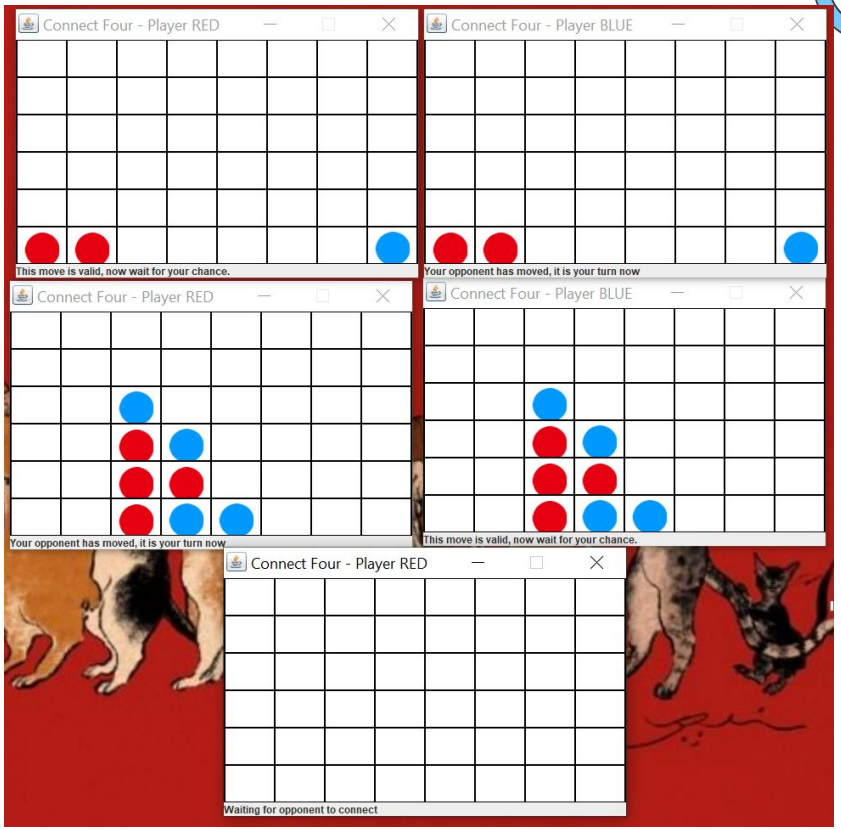



# Steps of implementation

**15. Handling for what happens if one player leaves and other stays.**

**16. Making the game scalable by implementing a few tweaks in the code so that the server can host more than one gameplays.**



# Resources!


- Connect-Four: Wolfram Mathematica
- How to play connect four board game
- Introducing threads in socket programming: G4G
- Implementing Java GUIs: oracle
- How to write a mouse listener: oracle

# Team Members

Kesha Bagadia

Yashvi Pipaliya

# Thank You