import java.io.\*;

import java.net.\*;

public class TicTacToeServer {

    private ServerSocket serverSocket;

    private Socket playerXSocket;

    private Socket playerOSocket;

    private PrintWriter playerXOut;

    private PrintWriter playerOOut;

    private BufferedReader playerXIn;

    private BufferedReader playerOIn;

    private char[][] board;

    private boolean isGameRunning;

    private char currentPlayer;

    public TicTacToeServer(int port) {

        board = new char[3][3];

        isGameRunning = false;

        currentPlayer = 'X';

        try {

            serverSocket = new ServerSocket(port);

            System.out.println("Server is running on port " + port);

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    public void acceptPlayers() {

        try {

            System.out.println("Waiting for Player X to connect...");

            playerXSocket = serverSocket.accept();

            System.out.println("Player X has connected.");

            playerXOut = new PrintWriter(playerXSocket.getOutputStream(), true);

            playerXIn = new BufferedReader(new InputStreamReader(playerXSocket.getInputStream()));

            System.out.println("Waiting for Player O to connect...");

            playerOSocket = serverSocket.accept();

            System.out.println("Player O has connected.");

            playerOOut = new PrintWriter(playerOSocket.getOutputStream(), true);

            playerOIn = new BufferedReader(new InputStreamReader(playerOSocket.getInputStream()));

            startGame();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    private void startGame() throws IOException {

        isGameRunning = true;

        sendToPlayers("Game starts! Player X goes first.");

        initializeBoard();

        while (isGameRunning) {

            sendToPlayers(printBoard());

            if (currentPlayer == 'X') {

                playerXOut.println("Your turn. Enter row (1-3) and column (1-3) separated by space:");

                processPlayerMove(playerXIn.readLine(), 'X');

            } else {

                playerOOut.println("Your turn. Enter row (1-3) and column (1-3) separated by space:");

                processPlayerMove(playerOIn.readLine(), 'O');

            }

            if (hasWinner() || isBoardFull()) {

                sendToPlayers(printBoard());

                sendToPlayers("Game over!");

                isGameRunning = false;

            } else {

                currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

            }

        }

        playerXSocket.close();

        playerOSocket.close();

        serverSocket.close();

    }

    private void initializeBoard() {

        for (int i = 0; i < 3; i++) {

            for (int j = 0; j < 3; j++) {

                board[i][j] = ' ';

            }

        }

    }

    private String printBoard() {

        StringBuilder sb = new StringBuilder();

        sb.append("\n");

        for (int i = 0; i < 3; i++) {

            sb.append(" ");

            for (int j = 0; j < 3; j++) {

                sb.append(board[i][j]);

                if (j < 2) sb.append(" | ");

            }

            sb.append("\n");

            if (i < 2) sb.append("---|---|---\n");

        }

        sb.append("\n");

        return sb.toString();

    }

    private void processPlayerMove(String move, char player) {

        String[] coordinates = move.split(" ");

        int row = Integer.parseInt(coordinates[0]) - 1;

        int col = Integer.parseInt(coordinates[1]) - 1;

        if (isValidMove(row, col)) {

            board[row][col] = player;

        }

    }

    private boolean isValidMove(int row, int col) {

        return row >= 0 && row < 3 && col >= 0 && col < 3 && board[row][col] == ' ';

    }

    private boolean hasWinner() {

        // Check rows, columns, and diagonals for winning condition

        // Implement your winning logic here

        return false;

    }

    private boolean isBoardFull() {

        for (int i = 0; i < 3; i++) {

            for (int j = 0; j < 3; j++) {

                if (board[i][j] == ' ') {

                    return false;

                }

            }

        }

        return true;

    }

    private void sendToPlayers(String message) {

        playerXOut.println(message);

        playerOOut.println(message);

    }

    public static void main(String[] args) {

        int port = 12345;

        TicTacToeServer server = new TicTacToeServer(port);

        server.acceptPlayers();

    }

}

SAVE this file as TicTacToeServer.java

import java.io.\*;

import java.net.\*;

public class TicTacToeClient {

    public static void main(String[] args) {

        String serverAddress = "localhost";

        int port = 12345;

        try (

            Socket socket = new Socket(serverAddress, port);

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in))

        ) {

            String serverMessage;

            while ((serverMessage = in.readLine()) != null) {

                System.out.println(serverMessage);

                if (serverMessage.startsWith("Your turn")) {

                    String move = userInput.readLine();

                    out.println(move);

                } else if (serverMessage.startsWith("Game over")) {

                    break;

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}

SAVE this file as TicTacToeClient.java

First compile both programs then first run server and after running server run the client in second terminal and then run client in another terminal as player 1&2

Server is running on port 12345

Waiting for Player X to connect...

Player X has connected.

Waiting for Player O to connect...

Player O has connected.

 