Tic-tac-toe server

Your task is to implement the client side of a tic-tac-toe server. The protocol, server, and game logic has already been implemented for you.

The code for this exercise is in lecture-code > evening_section > Lecture14 > tictactoe. You should only edit client.Client.java. You may use any of the objects defined in the client and protocol packages, but your code should not reference anything in the server package.

The Protocol

To get started, review all the code in the tictactoe package, focusing on the protocol sub-package. The protocol defines a set of accepted message types, represented by integer constants. A complete message between client and server will either just contain an integer indicating the message type or an integer followed by some data represented as a string. Message parsing has already been implemented for you in protocol.MessageParser.java. You can simply use the provided classes (Message, Protocol) to send messages in the expected format but here are some example messages in their raw format:

"19" - According to the protocol, 19 indicates the start of a game.

"20 0 1" - When the message contains more data, MessageParser looks for the message type at the start of message. In this case, the message type is 20, which indicates the user's turn. The server expects this message to be followed by two integers separated by a space, representing the square where the user is placing their piece. So, this message can be translated as "Place the human player's piece in square 0,1".

"20 01" - In this example, a turn has been requested but the rest of the message doesn't meet the formatting requirements. In this case, the server will return the following response:

"25 Invalid input! Enter your move in the format: row col. row and col must be 0, 1, or 2"

Implementing the Client

The client will need to get input from the user, which can then be sent to the server for processing. What the client requests from the user will vary depending on the server's response. Here are the basic steps you will need to implement.

- When the server sends a Protocol.QUIT method, the client should exit. This is already implemented.
- When the server sends either Protocol.YOUR_TURN or Protocol.INVALID_MOVE, the client should request input from the user to take a turn. When input is received, send it to the server as Protocol.MAKE_MOVE followed by the user input. Use the Message class to format and

send the message.

- When the server sends a Protocol.GAME_OVER message, the client should print the game outcome, which is included as the content of the server's message. Then, the client should allow the user to either continue the game or quit. If the user wants to continue, send a Protocol.START_GAME message. If the user wants to quit, send a Protocol.QUIT message.
- The user will find it helpful to see the board to decide where to make their move. To get a representation of the board, send a Protocol.SHOW_BOARD message. The server will respond with a Protocol.SHOW_BOARD message and a String representation of the board, which can be passed to Board.createBoard() to format it for printing.