Code Overview

The code offered is a straightforward Java implementation of a server-client Tic Tac Toe game. Two clients can join the game's server and play against one another. To regulate game flow and user involvement, the messages players exchange inside the game adhere to predetermined forms.

1. Classes and Their Functionality

There are five primary classes, and each one plays a particular part in the system.

1.1 TicTacToe.java

The logic behind the Tic Tac Toe game is implemented in this class. The game board, the current turn, and the logic to determine victory and draw circumstances are all included. Additionally, it offers ways to make movements, reset the board, and exchange turns.

Key methods:

- **reset()**: Resets the game board.
- **placeMove(int position, String marker)**: Tries to place the marker at the given position. If the operation is successful, it switches the turn to the other player.
- **checkWin()**: Checks for winning conditions for the current player.
- **checkDraw()**: Checks if the game is a draw.

1.2 ClientListener.java

This class is in charge of listening to incoming messages from the client and processing them appropriately. It extends Thread. On the server side, it manages the logic of the game, including verifying movements, examining the game's state, resuming the game, and sending messages to the adversary.

Key methods:

- **run()**: This overridden method from the **Thread** class is responsible for continuously listening to the client's messages and processing them.
- **processMessage(String message)**: Processes incoming messages, extracts the command or move, and forwards the message to the opponent if necessary.
- **restartGame()**: Restarts the game upon request from the client.
- **makeMove(String move)**: Attempts to make the provided move on the board.

1.3 ServerListener.java

This class extends Thread and prints messages by listening for responses from the server.

Key methods:

- **run()**: This overridden method from the **Thread** class listens to the server's responses and prints out the messages.

1.4 TicTacToeClient

By connecting to the server, establishing input and output streams, and launching a ServerListener to handle server answers, this class configures the client-side of the program.

Key methods:

- **main(String[] args)**: Sets up the client-side of the application, connects to the server, sets up input and output streams, and starts the ServerListener to process server responses.

1.5 TicTacToeServer.java

The server-side configuration of the program, accepting connections from two clients, and initiating the ClientListener for each client are all handled by this class.

Key methods:

- **main(String[] args)**: Sets up the server-side of the application, accepts connections from two clients, and starts the ClientListener for each client.

2. Formats for Messages

The system offers a variety of communication forms for diverse uses, including the following:

2.1 Connecting Message

A client connects to the server and then gets a message with the following formatting: "Connected to the server." and “Your marker is <marker> and your ID is <ID>.” their given marker (X or O), and their ID.

2.2 Move the Message

The syntax of a move request from a client is "move:<number>," where <number> denotes the desired location on the Tic Tac Toe board. For instance, "move:5"

2.3 Message for Restart

Clients can resume a game by sending the command "/restart".

2.4 Other Messages

Any communication that does not adhere to the aforementioned forms is categorized as a chat message and transmitted to the recipient client appended with the sender's ID.

3. Class Specifics and Implementation Details

Maintainability and scalability are supported by the object-oriented writing style and distinct separation of responsibilities used in the code. To prevent the program from crashing when meeting unexpected inputs or circumstances, exception handling has been incorporated where it is required. Additionally, using threads enables the server to manage several clients at once, resulting in a fluid gameplay experience.

3.1 TicTacToe.java

This class keeps track of an array board made up of nine Strings that correspond to the nine spaces on a tic tac toe board. Additionally, it keeps track of the turn variable, which is a String that represents the player's turn marker.

The checkWin and checkDraw methods check for winning and draw conditions, respectively, while the placeMove function tries to put a marker at a certain location on the board.

3.2 ClientListener.java

This class is in charge of processing messages that come in from the designated client. It continuously reads messages from the client and processes them using a while loop.

It initially determines if a message is a /restart message when it is received. If yes, it resumes the game if it has already been completed.

The class tries to move on the board at the designated position if the message begins with move:.

Any other message that isn't empty is regarded as a chat message and is sent to the other client.

3.3 ServerListener.java

This class is in charge of listening for answers from the server and publishing the messages. To decrease size, it only reads incoming messages and produces.

```
faruk.ulutas@Faruks-MacBook-Pro JavaSocketChat final % java TicTacToeServer
Usage: java TicTacToeServer <port number>
faruk.ulutas@Faruks-MacBook-Pro JavaSocketChat final % java TicTacToeServer
6000
Server started. Waiting for clients...
A client is connected, and it is assigned with the symbol X and ID=0.
A client is connected, and it is assigned with the symbol O and ID=1.
The game is started.
Waiting for Player O's move
Received O on 1
It is a legal move.
Waiting for Player X's move
Received X on 2
It is a legal move.
Waiting for Player O's move
Received O on 4
It is a legal move.
Waiting for Player X's move
Received O on 3
Received X on 3
It is a legal move.
Waiting for Player O's move
Received O on 10
It is an illegal move. (The move is out of the board.)
Received O on !
Invalid move. (The move should be a number.)
Received O on 55
It is an illegal move. (The move is out of the board.)
Received O on 1
It is an illegal move. (The position is already occupied.)
Received O on 7
It is a legal move.
Game over: Player X wins!
The game is restarted.
Waiting for Player X's move
```

```
Turn information: Waiting for opponent...
State of the board:
_________
O |X |__|
O |__|__|
__|__|__|
move:3
It's not your turn!

Turn information: Your turn!
State of the board:
_________
O |X |X |
O |__|__|
__|__|__|
move:10
Server says: "This is an illegal move. Please change your move!"
move:!
Server says: "This is an illegal move. Please change your move!"
move:55
Server says: "This is an illegal move. Please change your move!"
move:1
Server says: "This is an illegal move. Please change your move!"
move:7

Turn information: Waiting for opponent...
State of the board:
_________
O |X |X |
O |__|__|
O |__|__|
You win!
/restart
Game restarted. You have symbol O.
```

```
faruk.ulutas@Faruks-MacBook-Pro JavaSocketChat final % java TicTacToeClient
Usage: java TicTacToeClient <port number>
faruk.ulutas@Faruks-MacBook-Pro JavaSocketChat final % java TicTacToeClient 6000

Connected to the server.
Retrieved symbol X and ID=0.


Welcome to the Tic Tac Toe game!
To make a move, send a message in the following format:
move:<number>
where <number> is a number between 1 and 9 representing the position on the boar
d.
For example, to place your symbol at position 5, send 'move:5'.
_________
1 |2 |3 |
4 |5 |6 |
7 |8 |9 |
Remember, messages that don't follow the move format will be considered as chat
messages to your opponent.When the game is over, you can restart the game by typ
ing '/restart'.You can leave the game at any time by typing '/quit'.

Turn information: Waiting for opponent...
State of the board:
_________
__|__|__|
__|__|__|
__|__|__|
Selam
Client 1: Selam başlayalım mı?
Olur

Turn information: Your turn!
State of the board:
_________
O |__|__|
__|__|__|
__|__|__|
move:2

Turn information: Waiting for opponent...
State of the board:
_________
O |X |__|
__|__|__|
__|__|__|
```