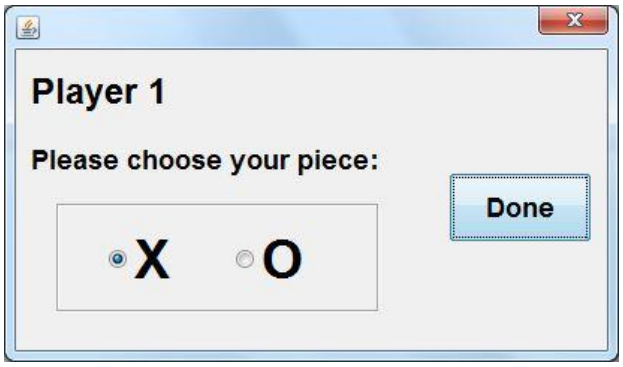
Introduction

This assignment is a Tic Tac Toe game application that can be played between two players over the network resided in two computers. This application was developed using Java programming language and utilizes socket programming as well as multi-threaded programming. It also enables the players to chat with each other while playing.

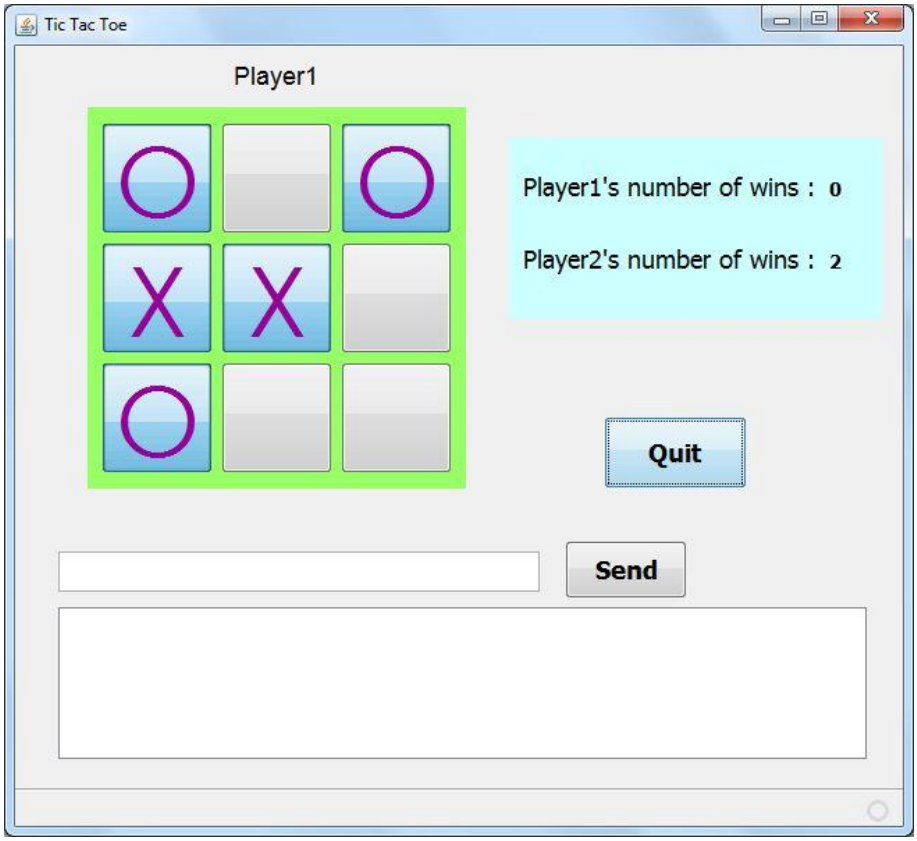
Instruction

After running the application, the user will first be asked to choose which player to play as, Player1 or Player2; Player1 is always the server and Player2 is client.

If the player chooses to be Player1 or server, (s)he has the privilege to choose the piece(X/O); then, Player1 will wait for Player2 to connect to it to start the game.



If the player chooses to be Player2, the application will ask for the server's (Player1) address to connect to. As soon as the connection is established, two players will observe the following screen and can start playing game and at the same time chat with each other using the bottom text field.



Each time a game is finished, a new game will start automatically.

Design and Implementation

This application consists of two main parts; the classes implemented for the gaming purposes and those implemented for the chatting purposes.

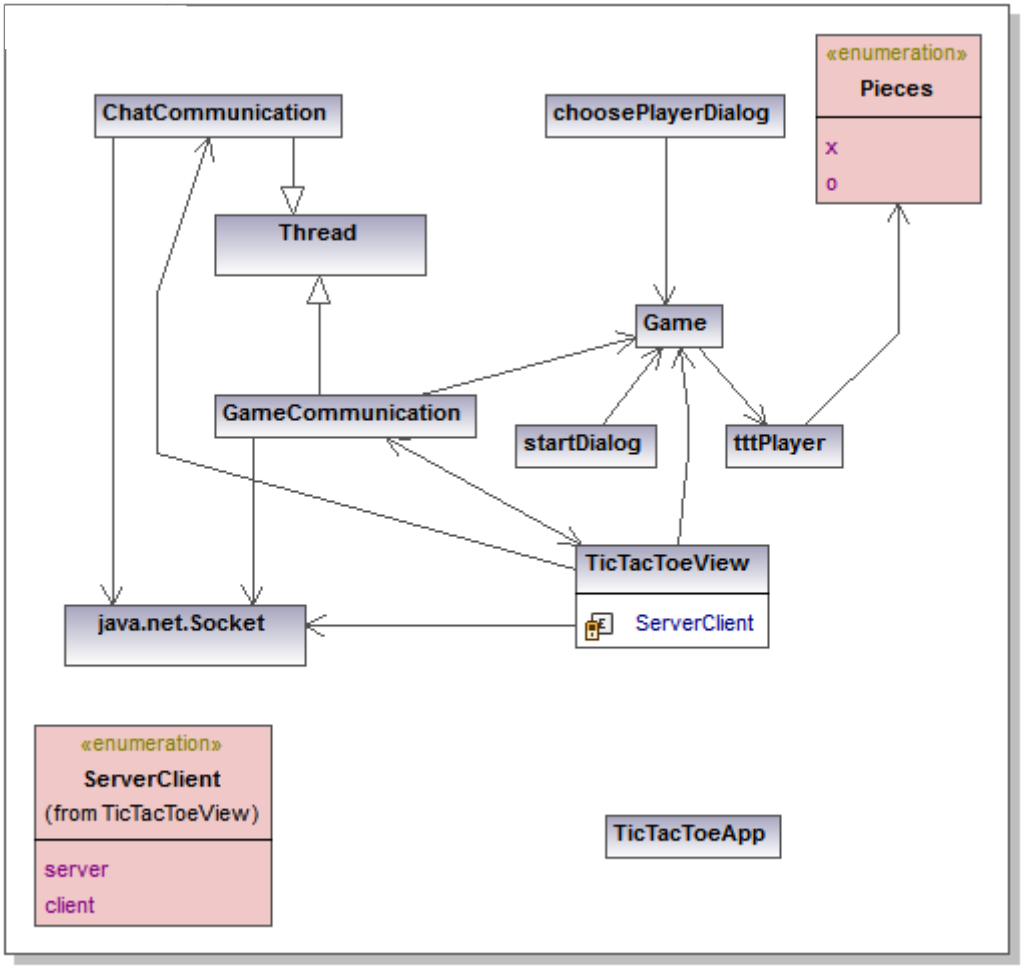
The game board tiles are represented using an ArrayList, each of which has a number from 1 to 9 assigned to. The game part comprises the following main classes:

- *tttPlayer*: contains the specifications of a player and their acquired tiles and number of wins.
- *Game*: contains the specifications of a game consisting of two *tttPlayer*, wining conditions, active player and the method to perform a move.
- GameCommunication: extends *Thread* and contains the routines to make the communication between two players for the gaming purpose.

The chat part comprises the following main class:

- ChatCommunication: extends *Thread* and contains the routines to make the communication between two players for the chatting purpose.

The relations between implemented classes are depicted here:



Socket programming and multi-threaded programming were exploited for this application whose aspects are going to be discussed in details next.

Socket Programming

After running the application, two sockets with two different ports will be allocated.

The first socket is used for players to chat with each other which is implemented in class *ChatCommunication*. The sockets are created in the main class *TicTacToeView* and are passed subsequently to *ChatCommunication* which is responsible for the message passing between players. The Player1 is always considered server and the Player2 is considered client.

The second socket is used for gaming and exchanging the moves of players which is implemented in *GameCommunication* class. Here also, the Player1 is always considered server and the Player2 is considered client. To inform another player about each move performed by current player, an integer between 1 and 9 representing the tile selected, will be send and the rest of processes needed to project this move will be taken care of by the main class *TicTacToeView* and class *Game*.

Multi-threaded Programming

In order to enable server sockets to keep receiving data while players are playing, and in order to have two sockets performing tasks explained above independently and without interruption, multi-threaded programming has been exploited. The classes *ChatCommunication* and *GameCommunication* are derived from *Thread* that makes them independent threads. In their *run( )* methods which specifies the task of the thread, only the receiving part or reading from socket is integrated; that way, the two sockets of chatting and gaming are always capable of sending and receiving data.