UNIVERSITY OF WINDSOR

SCHOOL OF COMPUTER SCIENCE

COMP-3670 COMPUTER NETWORKS

SUMMER 2019

Project Report

Implementing Tic-Tac-Toe Board game using Socket/TCP

Xiaoshuai Geng

Xiaoyue Liang

Yunfei Xia

Zekun Zhang

1. Introduction

The major task of this project is using Socket/TCP to implement a Tic-Tac-Toe board game. The whole project was coding in java 8. We used Object-Oriented idea to divide the task into several small tasks which are solved by implementing 11 classes in total. The application can be executed separately in two parts – the server program and client program.

1. Feature of Application

The server program and client program can be run independently, even on different PCs. Before starting game, the player should enter the server’s IP address and his name. After clicking start, the board will be ‘enabled’ and server will randomly decide player’s mark. By clicking the button on the board, a player can make his or her move. It is not allowed to make a move on a block which already been placed by a mark (the button will be disabled).

This Tic-Tac-Toe game is a human player opposing AI player game where an AI player will be run on Server and human player will have a visible Game board. The User can enter his or her name or choose restart or exit by clicking the button in menu bar.

The server is implemented using multi-thread method which allows different clients to connect at same time. Different client has its board and game saved in Server respectively.

1. Tools and technique

We used Java.net.socket as the tool for socket implementation. Server and Client build the connection through a socket of port (59090). Every time a client program wants to connect to a server, the client is required to enter server’s IP address.

We used Scanner for accepting input Stream and PrintWriter for output Stream. We pass all information about the game through this Stream and identify the type of the data by adding a header (inspired by IP packet header). The implementation in detail will be illustrated in next part.

GUI and java.awt is used for User Interface. The whole game board consist of 9 JButton. Each of them represents a single place slot for Tic-Tac-Toe game.

1. Implementation details

In Server program, we implemented all algorithms related to the Rule of the board game. That includes the implements of the model of 3x3 game board, 2 different kinds of player class including HumanPlayer and AiPlayer. A client will always have an Ai opponent which is run on server. Game file implements the algorithms about judging ‘win and lose’. The game class also take care of sending and receiving data command to/from the client. Each server can accept multiple client at same time by implementing multi-thread tasks.

The Client class is associated with GUI board. The Client class also process the data passed by server. In GUI board, we used 9 buttons filling up the board. We also implemented a menu bar which has restart and exit button. It has a text area at the bottom of the main interface, which allow player to enter his or her name. After press start button, the Tic-Tac-Toe will start.

A screenshot of a cell phone

Description automatically generated

Fig.1 Class diagram for Client program

The Client program will be executed by a test program. This test program will create a new Board, and the board will create a new client. The client will try to build a socket connection to Server class. The time client connects to the server, the server will ask for client’s name and then start the game and sending board information and AI player’s move.

The whole game is sequentially controlled by Server class, which means that the game board that displayed to player is only functional to receive and display the game board, sending player’s move, start, restart game, and exit the game. Each time the client starts the test program, only one board will be created. However, if the player does not exit program and click restart button, the original client will be closed, a new client with the same name will be restart. After the game has a result, or the player quit or restart, the original client will be disposed, the socket will be closed. The reason we implemented client in that way is since our client class is just an information sender and monitor. all the algorithms are implemented on server (including perform an AI player). Besides, Server is a multi-threads program, which allows several players connect and play simultaneously. To reduce server’s stress as much as possible, we are encouraged to dispose the client and close the socket if one game finished. Thus, we will not have least client program which takes up too much resources or taking up the socket forever.

A screenshot of a cell phone

Description automatically generated

Fig.2 Class diagram for Server program

The AI player algorithm is very simple. It has several preferred moves. Before finding a valid move, the AI player will try its preferred move first. It will keeps checking its move validity until the move can be placed.

The data Stream, that we used for server-client interaction, include 2 parts—the header and content. Both client and Server will detect the type of command. For instance, after accepting a client, a server will request for the name of the player by sending a request “NAME”. After receiving the command, the client will check the header using String.startwith() method and output its name to Server. By using this technique, we can easily pass different types of commands and information using only one socket. All the header will be listed below. (S for Server. C for Client)

|  |  |
| --- | --- |
| Sending informaton | Reacting |
| BORAD + 9 digits number (S) | Retrieving board info by using substring (C) |
| GAME + X/O/Draw (S) | Game end and get game result (C) |
| SYMBOL + X/O (S) | Get human player’s symbol (C) |
| NAME (S) | Send name back to server (C) |
| QUIT (C) | Close the socket （S） |

5.Conclusion

During the development, we faced a tough period of structural design due to inexperienced in Object Oriented program development. We changed and modified our structure for several times and reached out our best, resulting in this awesome application. It is a great opportunity for every group member to practice ourselves, and we tried our best to overcome all the difficulties. Each of our group members contributes significant work to the project. We do earn a lot of experience from the development.

6.

reference list:

Cs.lmu.edu. (2019). javanetexamples. [online] Available at: https://cs.lmu.edu/~ray/notes/javanetexamples/ [Accessed 23 Jul. 2019].