TJBB Final Project: Rock Paper Scissors

Temesgen Habte, James Kim, Brandon Hu, Bryant Nguyen

Table of Contents

Compiling and running

Server

1. Compile using:
   `javac Server.java ClientHandler.java`
2. Because the server is in a package, make sure to go back one folder and run using:
   `java TJBB.Server`

Client

1. Make sure python 3 is installed on the computer
2. For windows, just double click on Client.pyw and the gui should appear
3. For MacOS, run `python3 Client.pyw` in the terminal.

How-to Play Rock, Paper, Scissors

In the following sections, we'll walk you through how to start a server, how to connect to a server, how to create a game, and finally how to join and play a game.

Starting a Server:

Prior to starting the server, you need to make sure that Java JDK, JRE, and Python 3 are installed and able to run on your machine.

To start the server, simply run TJBB.Server (as said above) on the machine that you would like the server to run on. The server will then be open to communications on port 5000. There is no argument for the port so if it needs to be changed, it has to be changed in the code itself.

Since the current version of the client is hardcoded to attach to a specific server, you can change the connection IP to your server's IP address to allow the client to connect to your server. The same goes with the port number.

Starting a Client:

To start a connection with the server, run Client.pyw and the client will ask you to register with a username. If the username is already taken, they will be asked to change their registered username until they achieve a unique name.

If the username is found to be unique, you will then be registered within the server.

After Successful Registration:

After successfully registering with the server, you will be able to freely talk to all users in the server's chat channel, privately message other users, create a game for others to join, list available users for a game, challenge other users to a game of RPS, check the leaderboard, and unregister. Below are the respective text commands.

```
'/all <msg>'
'/message <username to msg> <msg>'
'/creategame'
'/games'
'/join <username to join>'
'/leaderboard'
'/unregister'
```

Creating a game
Users will be able to create a game by using the function '/creategame', opening a game for others to request to join.

Joining a game
In order to join a game, you must join a player with an available game. Available games are found using '/games'. Keep in mind, the challenged player has the right to reject your challenge / join.

Playing the game
Player1 will first be prompted to type "r" for rock, "p" for paper, and "s" for scissors. Then, Player2 will be prompted the same. Results will be displayed after.

Exit game

Players in a game may use command '/q' to forfeit and exit the game. To exit the program, the user simply closes the gui window or types '/unregister' and the program will unregister the user and close the program.

Other commands

'y' while being challenged will accept the challenge
'n' while being challenged will decline
'/q' while in a game to leave and forfeit the match
'/score' will retrieve the calling client's score (# of wins)

Rock Paper Scissors Network Protocol

The client and the server communicate over a socket. Strings are sent back and forth during the game for the commands and the outputs.The following strings are the commands that either the client and/or the server is expecting.

Register Username

To allow the user to pick a personalized username, the server first sends a message to the client asking to register a unique username. The client responds with their desired username.

If the username is taken by another user, the server will prompt the client to submit another username until a unique username has been entered and stored.

If the username is successful the server will notify the client by saying "That's a great username (desired username)!" and the client may now use the features of the program.

Messaging all users

'/all <msg>'

The client sends this message to the server, which the server sends to all other clients connected to the server.

Messaging one user

'/message <username to msg> <msg>'

The client sends this message to the server. The server then finds the username of the recipient and forwards the message as text to the client.

Listing available games

'/games'

The server receives this from the client, then searches through connected clients that are available for gameplay. The list is then sent back to the client.

Creating a game

'/creategame'

The client sends this message to the server when the user wants to start a game. The server sets the “inGame” boolean to false so that the user gets listed in the game list and other players can join. The server then sends back a message to let the user know their game has been created.

Joining available games

'/join <username to join>'

The server receives this from a client, then checks to see if that specified client is available to play still. It then sends a notification to that user to either accept or decline the client trying to join them.

Exit Game

'/q

The server receives this from one (or two) of the two clients in a game. Then after both clients choose their shoot selection (R, P, S, or /q) the server will determine whether or not one player wins by default due to the other playing quitting, or if the game results in a tie since both players quit. Server forwards results to both clients, changes clients' 'boolean inGame = true' property to 'false', and, if applicable, will increase the score of the winning player.

Score

'/score'

The server receives the command from the client, then prints only that specific client's win score.

List the Leaderboard

'/leaderboard'

The server receives the command from a client, then searches all users and sorts them by win count. This list is then forwarded to the client.

Unregister

'/unregister'

The server receives this from the client, then removes the client calling the function from a list of registered users. The client then closes marking the end of their session.

Gameplay

'r' or 'rock' while in game will send a "rock" to server
'p' or 'paper' while in game will send a "paper" to server
's' or 'scissors' while in game will send a "scissors" to server
An invalid input will result in a default loss (can occur for both players)
'/q' allows a player to quit mid-game and will result in a default loss (can occur for both players)

After clients choose their shoot selection, the server determines results and forwards those results to both clients, changes clients' 'boolean inGame = true' property to 'false', and, if applicable, will increase the score of the winning player.

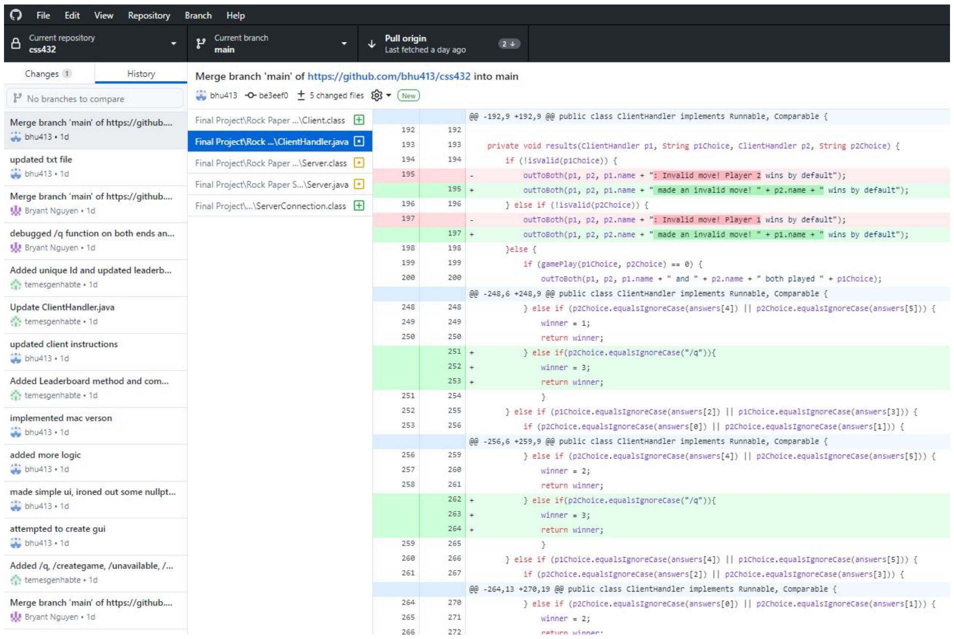
Personal Rubric

Criteria	Possible points	Desired Points	Reasoning
working application-level game	5.0 pts	5.0 pts	Our game is a working application-level game
required network architecture	5.0 pts	5.0 pts	Our game is a client-server network architecture
no crash in play	5.0 pts	5.0 pts	There was no crash in play even when testing multiple test cases
required feature - register	5.0 pts	5.0 pts	Unique ID registration is prompted at the start of client execution
required feature - list games	5.0 pts	5.0 pts	/games function will list available games (players open to a game)
required feature - create game	5.0 pts	5.0 pts	/creategame opens the current player's lobby to other players to join
required feature - join game	5.0 pts	5.0 pts	/join <username> allows a client to join the specified username's game if they have already created a game

required feature - exit game	5.0 pts	5.0 pts	/q allows for players to forfeit and quit their current game. Test cases included different inputs in combination with /q. Both players may quit to result in a tie.
Required feature - unregister	5.0 pts	5.0 pts	/unregister will remove the player from the registered players list
required feature - application specific protocol	20.0 pts	20.0 pts	Client -> server communication protocol is structured as: /<command> <payload>
bonus feature - chat	5.0 pts	5.0 pts	Our chat has 2 functions, private messaging to a registered user and a message to all. Functions /message and /all
bonus feature - scoreboard	5.0 pts	5.0 pts	/leaderboard will pull up all registered players and their wins in descending order of wins
bonus feature - game video	5.0 pts	0 pts	Not implemented
documentation	10.0 pts	10.0 pts	Our how to play provides a step by step and protocol is TBD
Best Game Bonus	10.0 pts	2 pts	We believe we got top 3 or higher compared to other groups due to completion, GUI, and lack of errors or crashes.

Total Points: 87/ 75

Proof of Version Control



Current repository: css432 · Current branch: main · Pull origin, Last fetched a day ago

Merge branch 'main' of https://github.com/bhu413/css432 into main

bhu413 · be3eef0 · 5 changed files

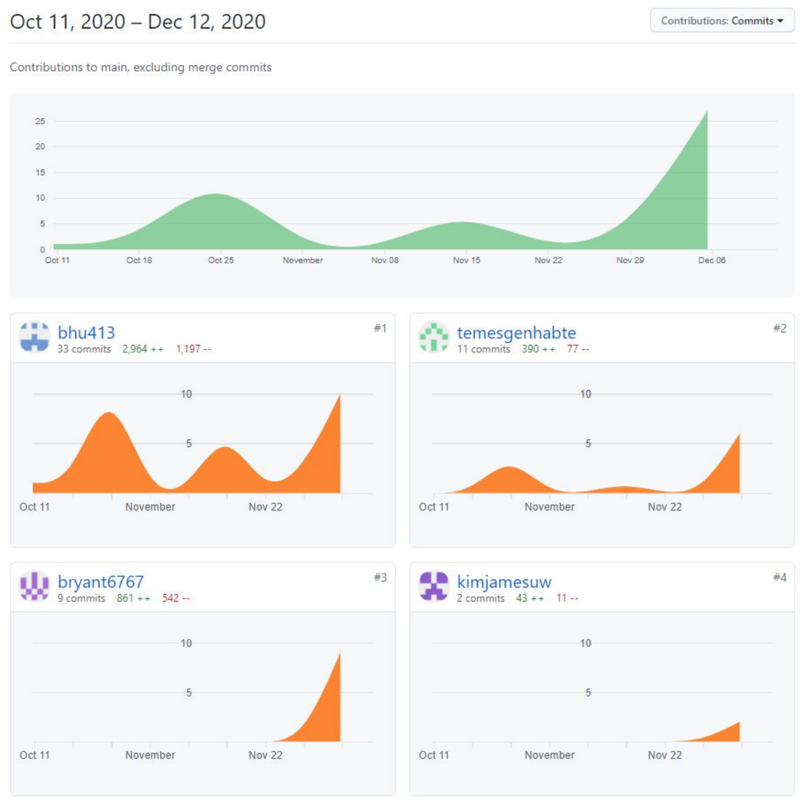
- Final Project\Rock Paper ...\Client.class
- Final Project\Rock ...\ClientHandler.java
- Final Project\Rock Paper ...\Server.class
- Final Project\Rock Paper S...\Server.java
- Final Project\...\ServerConnection.class

History:

- Merge branch 'main' of https://github.... — bhu413 • 1d
- updated txt file — bhu413 • 1d
- Merge branch 'main' of https://github.... — Bryant Nguyen • 1d
- debugged /q function on both ends an... — Bryant Nguyen • 1d
- Added unique Id and updated leaderb... — temesgenhabte • 1d
- Update ClientHandler.java — temesgenhabte • 1d
- updated client instructions — bhu413 • 1d
- Added Leaderboard method and com... — temesgenhabte • 1d
- implemented mac verson — bhu413 • 1d
- added more logic — bhu413 • 1d
- made simple ui, ironed out some nullpt... — bhu413 • 1d
- attempted to create gui — bhu413 • 1d
- Added /q, /creategame, /unavailable, /... — temesgenhabte • 1d
- Merge branch 'main' of https://github.... — Bryant Nguyen • 1d

```
@@ -192,9 +192,9 @@ public class ClientHandler implements Runnable, Comparable {

    private void results(ClientHandler p1, String p1Choice, ClientHandler p2, String p2Choice) {
        if (!isValid(p1Choice)) {
-           outToBoth(p1, p2, p1.name + ": Invalid move! Player 2 wins by default");
+           outToBoth(p1, p2, p1.name + " made an invalid move! " + p2.name + " wins by default");
        } else if (!isValid(p2Choice)) {
-           outToBoth(p1, p2, p2.name + ": Invalid move! Player 1 wins by default");
+           outToBoth(p1, p2, p2.name + " made an invalid move! " + p1.name + " wins by default");
        }else {
            if (gamePlay(p1Choice, p2Choice) == 0) {
                outToBoth(p1, p2, p1.name + " and " + p2.name + " both played " + p1Choice);
@@ -248,6 +248,9 @@ public class ClientHandler implements Runnable, Comparable {
            } else if (p2Choice.equalsIgnoreCase(answers[4]) || p2Choice.equalsIgnoreCase(answers[5])) {
                winner = 1;
                return winner;
+           } else if(p2Choice.equalsIgnoreCase("/q")){
+               winner = 3;
+               return winner;
                }
        } else if (p1Choice.equalsIgnoreCase(answers[2]) || p1Choice.equalsIgnoreCase(answers[3])) {
            if (p2Choice.equalsIgnoreCase(answers[0]) || p2Choice.equalsIgnoreCase(answers[1])) {
@@ -256,6 +259,9 @@ public class ClientHandler implements Runnable, Comparable {
            } else if (p2Choice.equalsIgnoreCase(answers[4]) || p2Choice.equalsIgnoreCase(answers[5])) {
                winner = 2;
                return winner;
+           } else if(p2Choice.equalsIgnoreCase("/q")){
+               winner = 3;
+               return winner;
                }
        } else if (p1Choice.equalsIgnoreCase(answers[4]) || p1Choice.equalsIgnoreCase(answers[5])) {
            if (p2Choice.equalsIgnoreCase(answers[2]) || p2Choice.equalsIgnoreCase(answers[3])) {
@@ -264,13 +270,19 @@ public class ClientHandler implements Runnable, Comparable {
            } else if (p2Choice.equalsIgnoreCase(answers[0]) || p2Choice.equalsIgnoreCase(answers[1])) {
                winner = 2;
                return winner;
```



The graphs show our project version control activity. The reason why Brandon's is so high is because this includes his entire main github folder (including homeworks, etc.) We used pair-programming via screen share to work out bugs as a team.