Rock Paper Scissors
-A Distributed Computing Project

By- Aparna Parida & Khadeeja Mastoor

Problem Statement

The project is based on the game of Rock-Paper-Scissors that most of us have enjoyed playing in our childhood.The effort here is to design a computerized version of the same using the client server paradigm.
The game has been built upon a 'single server and multiple client' model that has been achieved using 'socket connection' establishment between a client and the server alongside multithreading.
A GUI based program has been designed which provides a seamless experience through the multiplayer game. The details of all the previous duels can be viewed in the file that is maintained by the server and is updated regularly once a duel ends.

WorkFlow of the Game

As soon as the server and client programs are run, their respective interfaces pop up wherein the clients provide their choices.
There's a provision for the client to choose their opponent i.e. between an 'AI' and another client. Once the opponent and player selection are made, the connection can be established by clicking on a button.
The player and opponent selection can be cross checked with the server side once the connection establishment is complete.
After the sockets are connected via their IP address and their port number, the players can move forward and make their selection between the three available choices of rock, paper and scissors.
The rules of the game are as follows:

1. Scissors cuts the paper to win
2. Rock smashes the scissors to win
3. Paper covers the rock to win

The selection for any of these choices can be made till the player clicks on the 'Quit' button after which the client submits the scores at the server and closes the connection from the server.
Once a client is removed the server updates the file (that records the history information of the duels) with the details of the duel that was played.
This file can be viewed by clicking on the ‘GamePlay’ button on the server side interface.
A clearer picture of the proceedings can be seen in the attached Use-Case diagram.

Elements Associated

Details of each element on the Server side interface:

1. **GamePlay**: This button directs the user to the file that records the details of all the previous duels that were held along with date and time information of the duel.
2. **TextArea**: The text area at the server side gives an insight to the user of the proceedings of the duel.For example, the details of the currently available players, player selections and the winner information.
3. **Quit**:This button lets us quit the server. As a result, no further duels can happen between clients.

Details of each element on the Client side interface:

1. **Your Name:**This textfield lets the user enter his/her name.
2. **Your Opponent’s Name:**This textfield lets the user enter his/her opponent’s name.
3. **Click to Start Server Connection:** This button when clicked establishes a socket connection of the client with the server via ip address and port no.
4. **Rock:**Button for player selection corresponding to Rock.
5. **Paper**:Button for player selection corresponding to Paper.
6. **Scissors**:Button for player selection corresponding to Scissors.
7. **Text Area**:The text area prompts to make a player selection and as soon as the selection is made by both the clients is displays the winner.
8. **Quit Server Connection**:This button lets the user to close the socket of the corresponding client and exit from the game.
9. The labels below the ‘Quit’ option showcase the scoreboard of the match. The names of the labels are updated once the Connection to the Server has been initiated.

Classes Present

On Client side:

public class Client

This class implements ActionListener and extends Frame which are the required functionalities for creating the GUI of the game. ActionListener is essential to implement actions with button clicks which is further implemented by associating a command action with respective buttons. The frame makes use of BoxLayout for setting elements in the interface and is being initiated using a constructor in the class.

This class further declares two constructors as follows:

a. public Client (Frame f) -This constructor takes the frame as an argument and creates a frame. It sets a defined size for the frame and also sets up the location on the screen along with assigning various graphic colors to the respective elements. In the beginning, the player selection buttons, i.e. 'Rock', 'Paper' and 'Scissors' has been disabled until the Connection to the Server has been made. Once the Connection is established, the 'Click to Start Server Connection' button is being disabled.
b. public Client (String player_nick, String opponent_nick, String server_name) - This constructor is used to create a socket connection and receive the messages from the client and send messages to the client. It sets up the BufferedReader and PrintWriter that is essential for message passing between the server and the client through the defined port. In the GUI, the need for BufferedReader has been omitted because the input is being obtained through the TextFields and Button click.

Following methods are declared within the class:

a. public void actionPerformed (ActionEvent e) - This method takes the object of ActionEvent as a parameter and defines the event that should occur upon each button click. A command has been declared with each button click which is then used to determine the corresponding actions that needs to be performed. In this method, with the 'Quit' button, an associated operation has been set where the Clients sends its respective number of Wins to the Server upon connection termination.
b. public static void increp1 () - This method increments the value of counter whenever the player wins.
c. public static void increp2 () - This method increments the value of counter whenever the opponent wins.

d. public static void Appedta (String txt) - This method is used to append text into the textArea whenever a message needs to be displayed to the Client. All the Command Line messages are also being directed onto the textarea using this method.

Following Classes are declared within the class:

a. class W1 extends WindowAdapter - This class is used to add a closing function to the frame. System.exit(0) is executed as a result of this.
b. class MessageThread extends Thread - This class, and hence the thread is initiated whenever the Client initiates player selection by clicking on either of the three selection buttons, i.e. 'Rock', 'Paper' or 'Scissors'. After that, the thread runs an infinite loop until the user clicks on exit button. Within this loop, the thread listens to the client's selection and then sends the same to the Server. On the Server side, upon receiving the opponent's selection, the duel occurs and then the result of the duel is replied back to the client which is listening it on the thread.

On Server Side:

public class Server

This class implements ActionListener and extends Frame which form the building block for creating the GUI for the game on the Server end. The GUI consists of three components, namely a textArea upon which the messages and instructions on the Server side are displayed, a 'GamePlay' button that opens up the file where the data of the duels is stored and a 'Quit' button that closes down the server. The actionListener is required to add the action to a particular button.

Following methods are declared within the class:

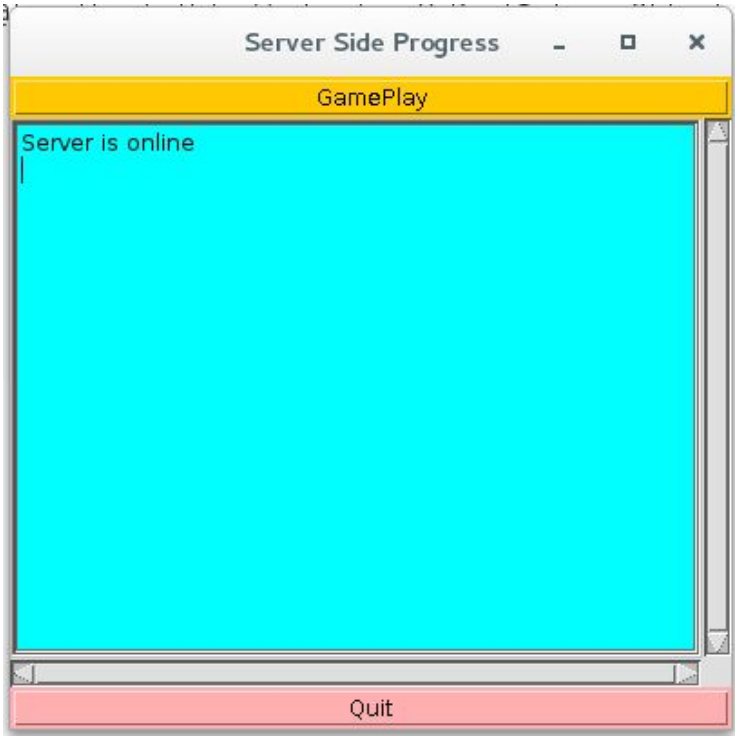
a. private void createServer () - This method creates the frame for the GUI of the game on the server's end and then sets up BufferedReader and PrintWriter that is essential for the message passing between the Server and the Client. This method also establishes the socket connection and then accepts the client that is trying to connect in the defined port.
b. public void actionperformed (ActionEvent e) - This method takes the object of ActionEvent as a parameter and defines the event that should occur upon each button click. A command has been declared with each button click which is then used to determine the corresponding actions that needs to be performed. The operation for 'GamePlay' button and the 'Quit' button has been declared in this step.

c. public static void Appedta (String txt) - This method is used to append text into the textArea whenever a message needs to be displayed to the Server. All the Command Line messages are also being directed onto the textArea using this method.
d. Public static void appendtofile(String fileName, String str) - This method is used to append the result of a duel to the file 'RockPaperScissors_data.txt'.
e. private void duel(String player_nick) - This method performs the duel of the player with 'AI' when the client opts to play versus computer.
f. private void duel(String player_nick, String opponent_nick) - This method performs the duel of the client with the player they have opted as their opponent.
g. private static String playerWinner(PlayerService player_service, String choiceAI) - This method returns the winner between the player and the 'AI'.
h. private static String playerWinner(PlayerService player_service, PlayerService opponent_service) - This method returns the winner between the two clients.
i. private static String drawServerSelection() - This method uses a random function to find the selection for the server and then based on that, the duel with the client occurs when the client opts to play against the computer.

Following classes are declared within the class:

a. class W1 extends WindowAdapter - This class is used to add a closing function to the frame. System.exit(0) is executed as a result of this.
b. class PlayerService- This class implements Runnable interface to achieve multithreading. This class creates the variables that are used further on in some associated methods:
    1. private PlayerService(Socket client) - After the variables have been created, this method is used to start the thread when the Server receives the data about the connected Client and their chosen opponent. Within run(), the loop then listens for the Client's selection and checks for the player the client has chosen as their opponent. When all the required conditions are met, the duel is instantiated and the result of the duel is replied back to the clients.
    2. private void sendMessage(String txt) - This method is used to send the message from the Server to the client.
    3. private String playerReturns() - This method returns the respective player name from the object associated with it.

GUI


Server Side Progress
GamePlay
Server is online
Quit


Rock Paper Scissors
By- Aparna & Khadeeja
Your Name
Your Opponent's Name
Click To Start Server Connection
Rock
Paper
Scissors
Client Progress...
======$#%|DC-PROJECT|%#$=======
| Rock Paper Scissors |
==========**<^>**==========
Input Into the Fields Provided.
When you are done, press the Button follow
Quit Server Connection

Use-Case diagram