Lab 2 : Distributed Systems

Nudrat Nawal Saber
1001733394

Environment:

Project was developed on

- Java Platform
- Apache Netbeans IDE v.12.0.0
- Windows 10 Home

Execution Guidelines:

1. Open project with folder name **SocketCheck2** on the IDE .

2. The project consists several .java files where running Server.java and Client.Java are only necessary for execution.

3. Start with running the sever.java file by right clicking on the file from project window and clicking run (can use shift+f6 shortcut).

4. After successful run the server window will appear where connected clients will show the active clients after a successful connection also it will update if a client disconnects.

5. Now run the client.java file similarly as stated in step 3 using (shift+ f6). Successful execution will result in a client window where we will be asked to provide a user name. As this is the first client, run any input name can be accepted.

6. After providing a name clicking on connect to server will establish a connection with the server and show a client window with connection information, a queue, a text box to write a text which can be send to server for lexicon check by pressing the send button and a text box to make additions to the server's lexicon.

7. Words entered by the client will be placed into a queue and will be shown. Every 60 seconds, the server will poll each client's queue. It'll show the message in server saying: " Server is polling. Queue is empty. No words sent to server. Polling Completed ".If the queue is not empty, the server will retrieve the contents of the queue and add those to its lexicon and will result in a server message: "Server is polling. Following words sent to server.[lexicon].Polling Completed".

8. Once the client has been polled, the contents of the queue will be purged. The server will remove any duplicate entries in the lexicon. Subsequent comparison between user-supplied text files and the lexicon reflects the updated contents of the lexicon.And also Client maintains a FIFO queue
.
9. Providing no text input in the text box and clicking send button will result in a sever error message:

   'Enter text before sending to server'.

10. Providing with valid text, the server will return lexicon checked sequence. For example, assume a text file with this text was uploaded to the server:

    the quck brown fx jumpz over the lazy dg

    If the server had a lexicon with the following contents:

    quck jumpz dg fx

    The server should return a a text file with the following contents:

    the [quck] brown [fx] [jumpz] over the lazy [dg]

11. The lexicon file is names as lexicon.txt where the current provided data are ' quck jumpz dg fx hllo wrld' .And data will be added if further sent from the client and the lexicon file will become updated. For example updating the lexicon.txt with a new content bx which represents box and sending the line 'black bx' will result in 'black [bx]'.

12. Now if the client.java is again executed while the previous one is running it will open a new window and option to enter user name. In this case providing same user name as the previous client will return an error message saying to change the user name as it has already been created, and refresh the provide name window.

13. More than 3 clients can run simultaneously and all the connected client name should be visible in the server window. If any client exits using the exit button the server will update the connected clients data.

Limitations:

Currently all the features are working as prescribed in the project outline.

Citation:

1. https://www.javatpoint.com/socket-programming
2. https://www.geeksforgeeks.org/multithreaded-servers-in-java/
3. https://docs.oracle.com/javase/tutorial/networking/sockets/
4. https://docs.oracle.com/javase/tutorial/networking/sockets/clientServer.html
5. https://docs.oracle.com/javase/tutorial/networking/sockets/readingWriting.html