

Programming Assignment #1 Report

Personal information of each team member: Full name, UF ID, and Email

- Kent Phipps
 - kent.phipps@ufl.edu
 - 4020-9184
- Waleed
 - waleed.aref@ufl.edu
 - 9618-5607
- Simar Khetpal
 - s.khetpal@ufl.edu
 - 6221-7796

How to compile and run your code

In order to compile and run the code, you will first need two different machines that each have the Client and Server files with jokes. The next step would be to compile and run the server program, for which you need to navigate to the folder where the server program "Server.java" is located using a terminal window within your compiler. Once you're in the Server's folder, compile the "Server.java" file by typing the command "javac Server.java." Once that has been compiled, you can run the Server program by typing "java Server <port>," but you need to replace ``<port>`` with the appropriate port number to use for the server's connection.

After this has been completed successfully, the next step is to compile and run the client program. To do this, open a terminal window on your second machine and navigate to the folder containing the Client program "client.java." In the Client's folder, compile the "client.java" file by executing the command "javac Client.java." To run the Client program, type "java Client <ServerIP> <port>," replace "<ServerIP>" with the IP address of the server, and replace "<port>" with the port number used when starting the server program.

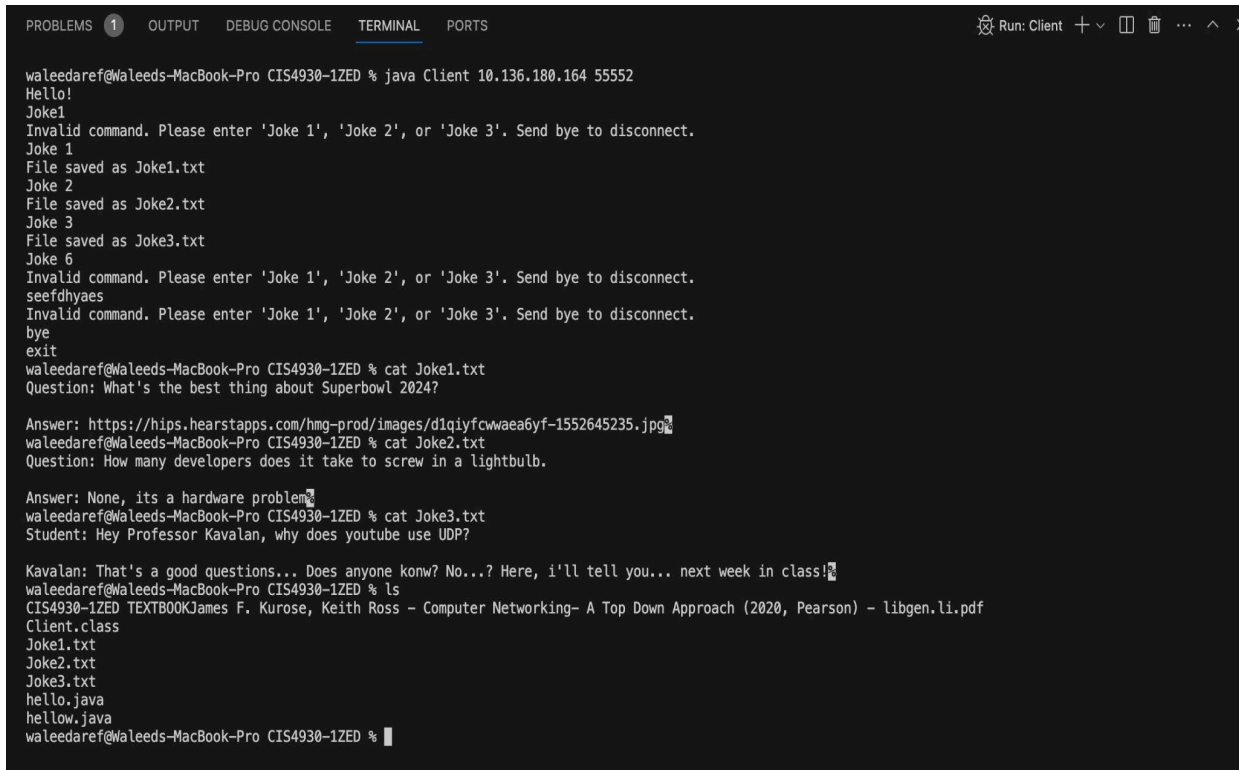
Finally, with this done, you can execute commands as both the Client and Server are running. Execute commands from the Client's terminal (second machine) to interact with the Server. To request a joke, type one of the following commands: "Joke 1", "Joke 2", or "Joke 3". The server will respond by sending the corresponding `joke.txt` file back to the Client. Once you are ready to terminate the client-server connection, type the command "bye" from the Client's terminal (second machine) to close the connection between the Client and Server.

Description of your code structure

The purpose of the client program is to connect to a server using socket communication over TCP/IP. Thus, our code essentially starts by initializing a "Socket" object to establish a connection with the server, specifying the server's IP address and port number. The client also sets up "BufferedReader" for reading input from the user via the console and "DataOutputStream" as well as "DataInputStream" for sending and receiving data to and from the server, respectively. Once it has successfully connected to the server, our code prompts the client to receive a greeting message ("Hello!") from the server, which is printed out. The client then enters a loop, allowing the user to input commands ("Joke 1", "Joke 2", "Joke 3", or "bye"). These commands are then sent to the server, and the client waits for a response. From here, if a joke command is sent, the server responds with the content of the corresponding joke file, which the client then saves to a local file. If "bye" is sent, the server acknowledges by sending a "disconnected" message, prompting the client to exit the loop, close all connections, and terminate.

Moving forward to our server program, which works to listen for connections on a specified port and handle requests from clients. Our code first initializes a "ServerSocket" to accept incoming client connections. Once a client connects, the server sets up "DataInputStream" and "DataOutputStream" to receive data from and send data to the client, respectively. The server then sends a "Hello!" message to the client upon establishing a connection. Afterward, it enters a loop, listening for commands from the client. Based on the received command, the server's action can vary as it can do the following: 1) fetches and sends the content of the following requested joke file ("Joke 1", "Joke 2", "Joke 3"), 2) send an error message for invalid commands, or 3) send a "disconnected" message and break the loop if "bye" is received. Thus, overall, the server is designed to send file contents as strings over the socket connection.

Show some execution results



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS Run: Client + - [ ] [ ] ... ^ x

waleedaref@Waleeds-MacBook-Pro CIS4930-1ZED % java Client 10.136.180.164 55552
Hello!
Joke1
Invalid command. Please enter 'Joke 1', 'Joke 2', or 'Joke 3'. Send bye to disconnect.
Joke 1
File saved as Joke1.txt
Joke 2
File saved as Joke2.txt
Joke 3
File saved as Joke3.txt
Joke 6
Invalid command. Please enter 'Joke 1', 'Joke 2', or 'Joke 3'. Send bye to disconnect.
seefdhyaes
Invalid command. Please enter 'Joke 1', 'Joke 2', or 'Joke 3'. Send bye to disconnect.
bye
exit
waleedaref@Waleeds-MacBook-Pro CIS4930-1ZED % cat Joke1.txt
Question: What's the best thing about Superbowl 2024?

Answer: https://hips.hearstapps.com/hmg-prod/images/d1qiyfcwaea6yf-1552645235.jpg
waleedaref@Waleeds-MacBook-Pro CIS4930-1ZED % cat Joke2.txt
Question: How many developers does it take to screw in a lightbulb.

Answer: None, its a hardware problem
waleedaref@Waleeds-MacBook-Pro CIS4930-1ZED % cat Joke3.txt
Student: Hey Professor Kavalan, why does youtube use UDP?

Kavalan: That's a good questions... Does anyone konw? No...? Here, i'll tell you... next week in class!
waleedaref@Waleeds-MacBook-Pro CIS4930-1ZED % ls
CIS4930-1ZED TEXTBOOKJames F. Kurose, Keith Ross - Computer Networking- A Top Down Approach (2020, Pearson) - libgen.li.pdf
Client.class
Joke1.txt
Joke2.txt
Joke3.txt
hello.java
hellow.java
waleedaref@Waleeds-MacBook-Pro CIS4930-1ZED %
```

Lesson learned

While class lectures give us the opportunity to learn about TCP/IP communication, working on a project that exemplifies the concepts allows us to showcase what we learned as well as show us what we might not fully understand yet. Thus, through this assignment, we were able to learn more of two things. Firstly, the basics of socket programming and network communication were taught. This entails understanding how data is transmitted between machines through sockets, the significance of port numbers, and more details about the TCP/IP protocol suite. Our experience in setting up server and client connections provided us with valuable insights into establishing network links and managing data exchanges effectively. In addition to this lesson, we also learned more about the importance of exception handling and network applications. We dealt with various challenges, including network errors, incorrect user inputs, and file access issues, which eventually led us to build more resilient systems that are capable of handling unexpected events and maintaining functionality in the proper way. One specific example of this was when we encountered an issue of the server's inability to recognize a "bye" command from the client intended to signal a disconnection. Initially, we attempted a temporary workaround by simply echoing the "bye" message from the client side without properly terminating the connection. However, afterward, inspired by a team member's insight

and initiative, the cause of the miscommunication was identified, and the correct solution was implemented, which allowed for proper disconnection as originally intended. Thus, overall, this experience not only enhanced our technical skills in Java socket programming but also deepened our understanding of the functionality of network applications.

Any additional comments

N/A