Minh Nguyen

02/28/2020

The first thing I did in my code is initialize all the variable I need for my code as well as import all the libraries I would need as well. In the constructor for my ChatServer class, I made a new server with the socket port (8888). This will be the port that I use for my server. I then put a while loop (lines 25-29) so that it will run for as long as the server is on, while the server is running, the code will accept any incoming socket and make a new thread with that socket using my UserThread class.

In my UserThread class constructor, I initialized the data output/input streams. In the run function of the class, I made a while loop that while the server has not stopped, I would ask the user for a username (String UserMssg, lines 56-67). I then if the hashmap containing a list of all current users (ConcurrentHashMap users) already had a username already. If it did, the loop would begin again and if not, then the loop would put a new entry in the hashmap with the socket(key), and username(value). Then the code would call the SendToAll() method (lines 111-121). The SendToAll would cycle through the hashmap and send a message to all users. In this case it would be to announce that a new User has joined. After the Server sent the new user message to every other user, I would then go into another loop and continue to take input from the user as long as the user did not type "/quit". If so, the Server will send a message saying "bye bye [username]" to all other users and then delete the user from the list of users and close the connection. The server will then check if the list is now empty. If so then the Boolean that represent if the server has stopped yet will be true. This will shut down the server(*). Alternatively, if the user type "/allusers", the server will send only that user a list of current connected users.

In my server class, every thread is responsible for their own read and write. I took this approach because I believe unlike the client class, there is no way a user would have need to read and write concurrently because they could not happen at the same time.

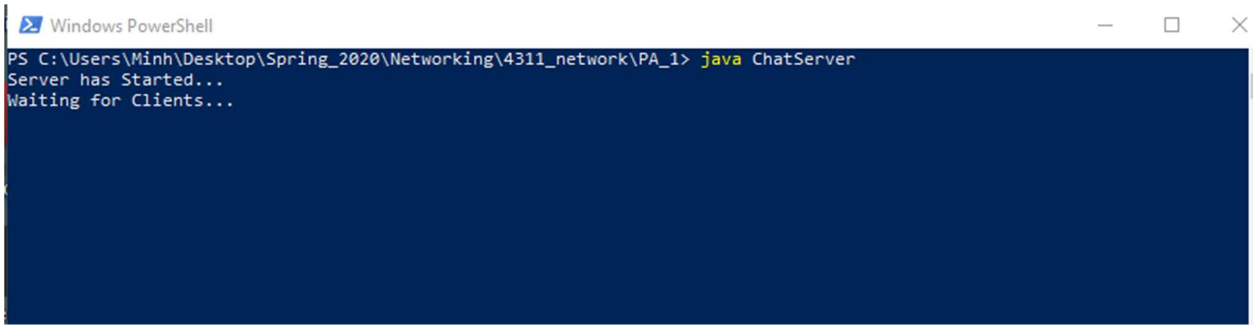
My client class as mentioned above has two threads, one for reading and one for writing. The reason I did it this way is because while the user is writing their message, the client should be continuously be taking input from server about other users' messages and posting it onto the chat.

My client class constructor implements the I/O streams as needed as well as start both of the reader and writer threads. The reader thread (lines 33-44) takes any incoming messages from the DataInputStream from the server (ie. Other user's messages) and print them to the console. The writer thread (lines 49-72) takes all input from the keyboard and evaluate the string. If the string is "/quit" close all connection and shut the client down. If the String is any other message, then the client will send that message through the output stream to the server.

* That was just for testing purposes, I understand that the case where one user join and then leave would shut down the entire server, preventing any other user from joining. I only included for testing purposes.

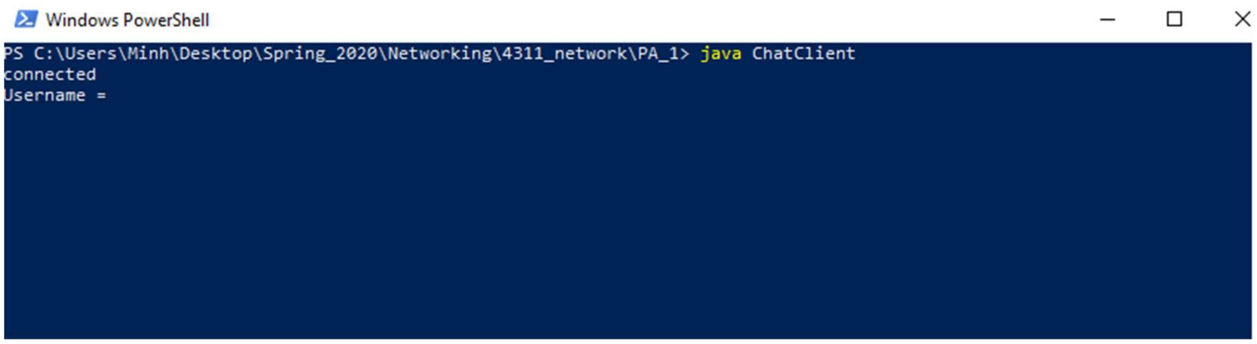
Screen Shots:

server starting


Windows PowerShell

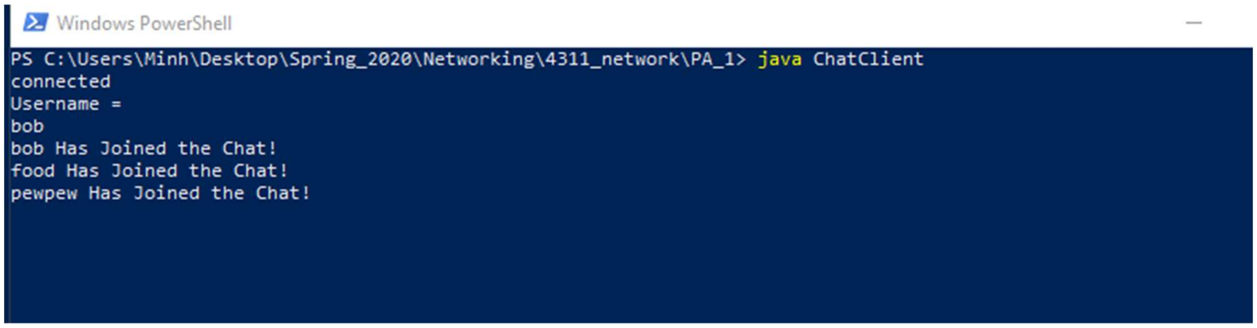
```
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1> java ChatServer
Server has Started...
Waiting for Clients...
```

Client 1 asking for username

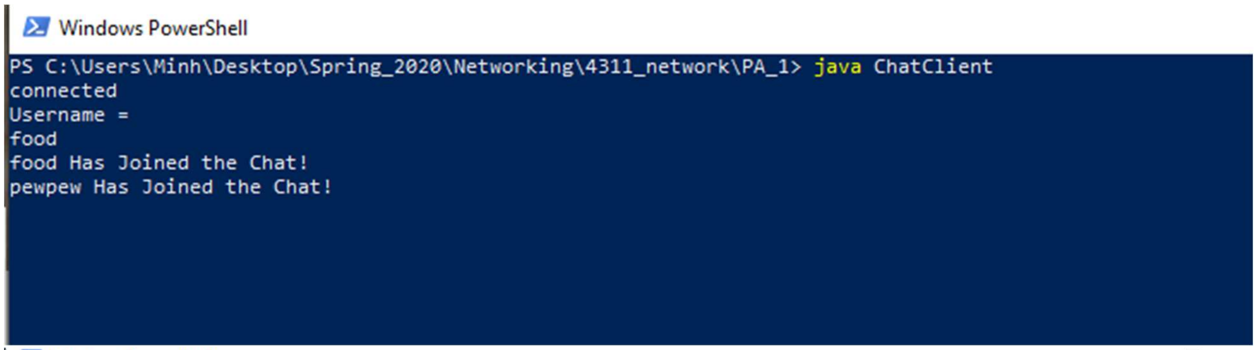


```
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1> java ChatClient
connected
Username =
```

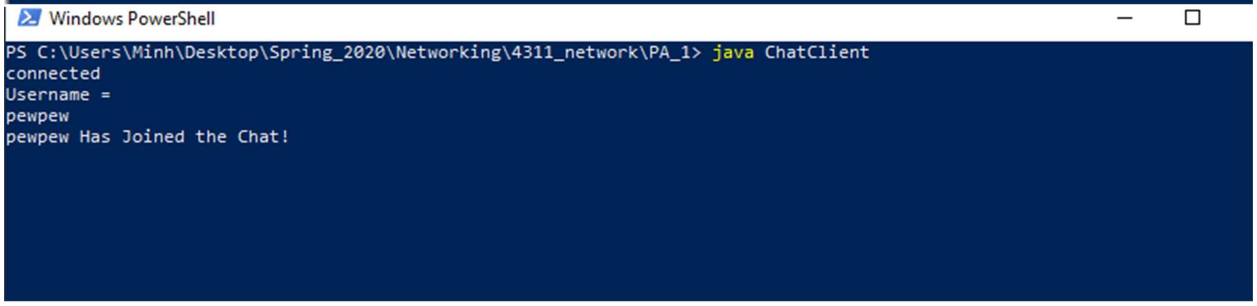
Client 1 ,2 ,3 (aka bob, food, pewpew) joining the chat



```
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1> java ChatClient
connected
Username =
bob
bob Has Joined the Chat!
food Has Joined the Chat!
pewpew Has Joined the Chat!
```

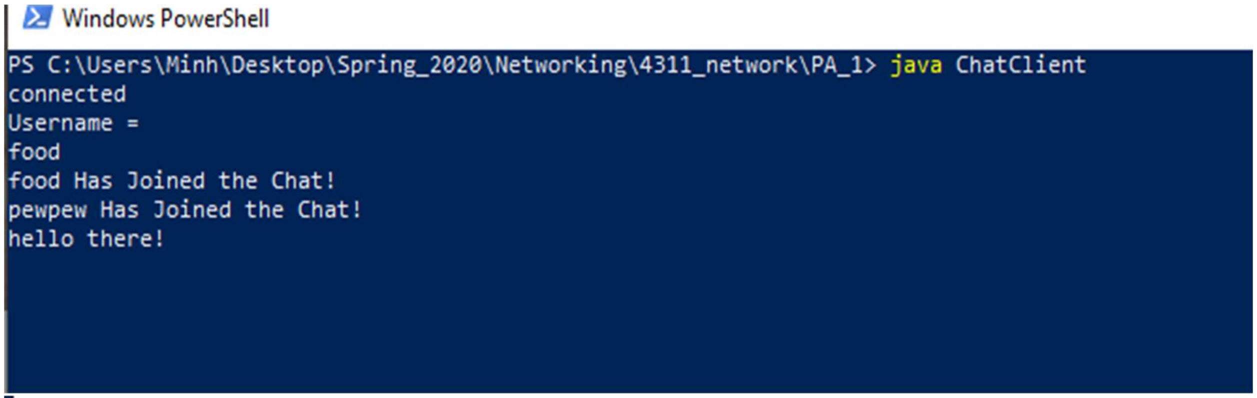


```
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1> java ChatClient
connected
Username =
food
food Has Joined the Chat!
pewpew Has Joined the Chat!
```



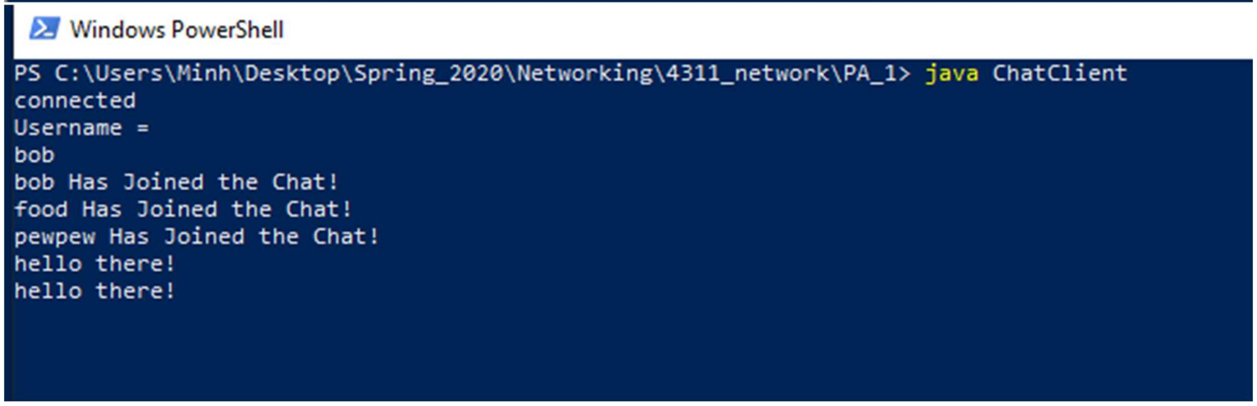
```
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1> java ChatClient
connected
Username =
pewpew
pewpew Has Joined the Chat!
```

Clients input/output “hello” test:



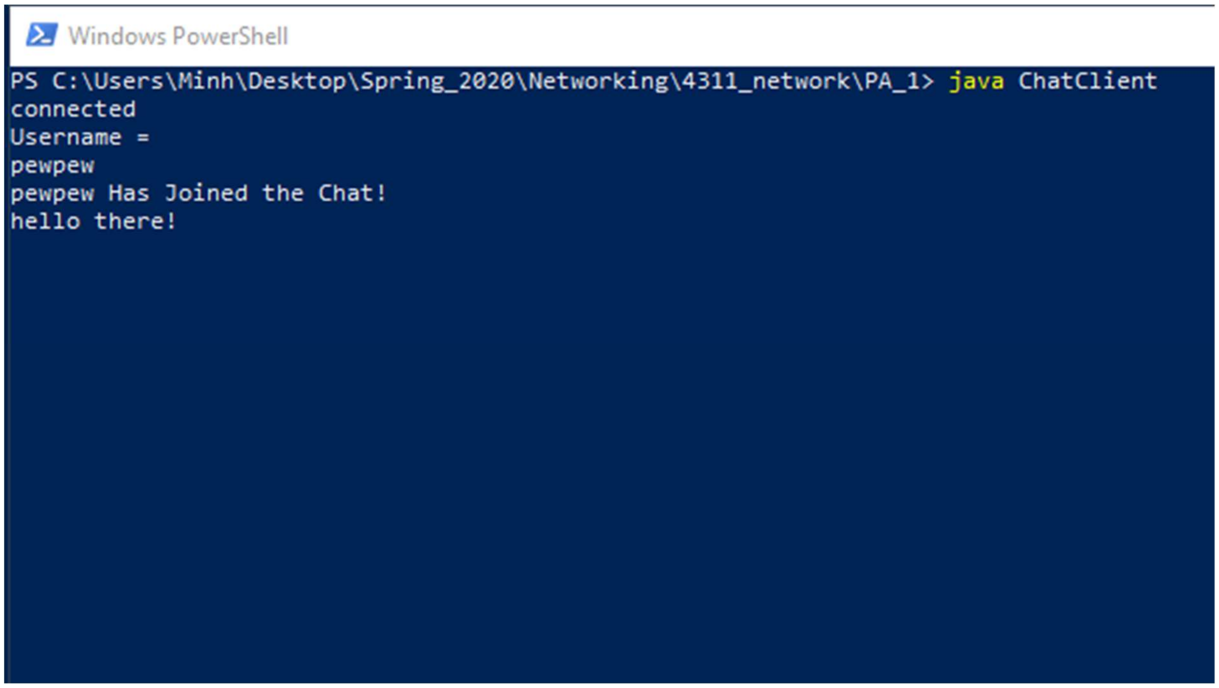
Windows PowerShell

```
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1> java ChatClient
connected
Username =
food
food Has Joined the Chat!
pewpew Has Joined the Chat!
hello there!
```



Windows PowerShell

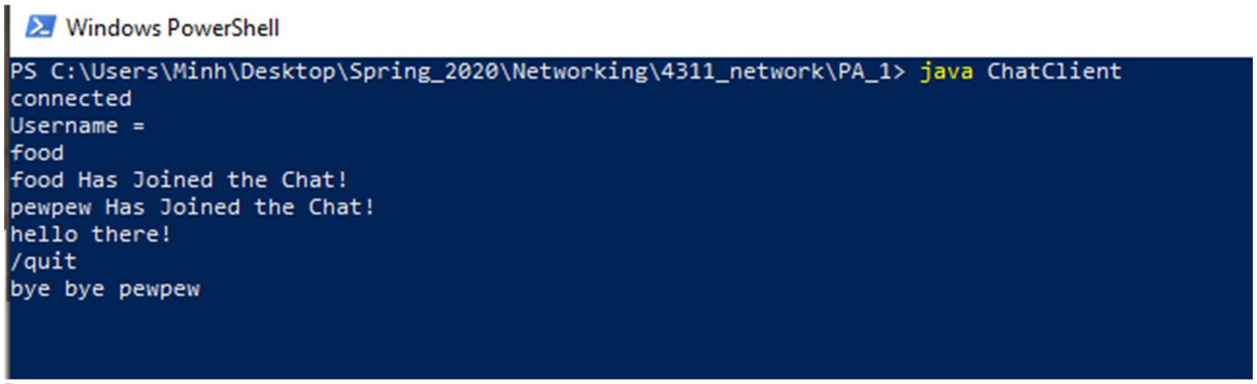
```
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1> java ChatClient
connected
Username =
bob
bob Has Joined the Chat!
food Has Joined the Chat!
pewpew Has Joined the Chat!
hello there!
hello there!
```



Windows PowerShell

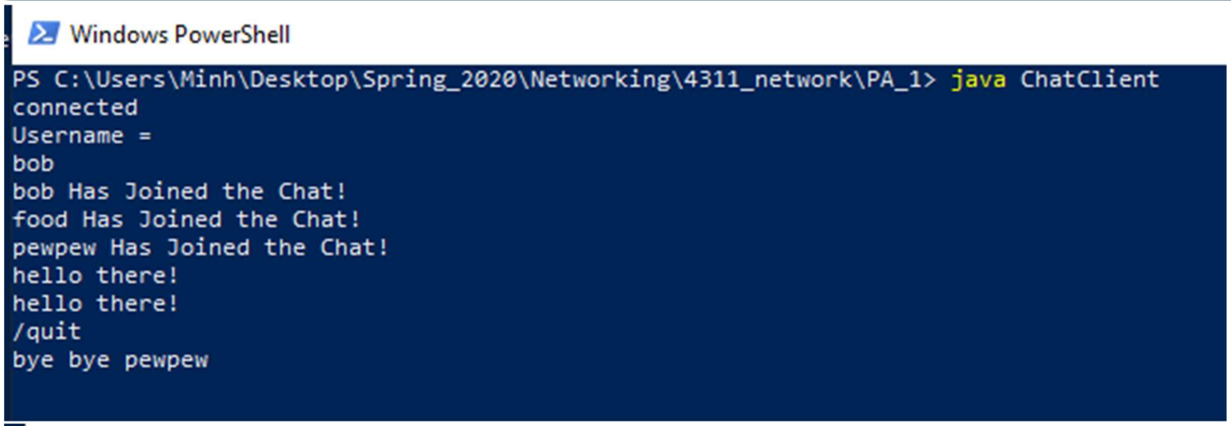
```
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1> java ChatClient
connected
Username =
pewpew
pewpew Has Joined the Chat!
hello there!
```

Client goodbye pewpew test:



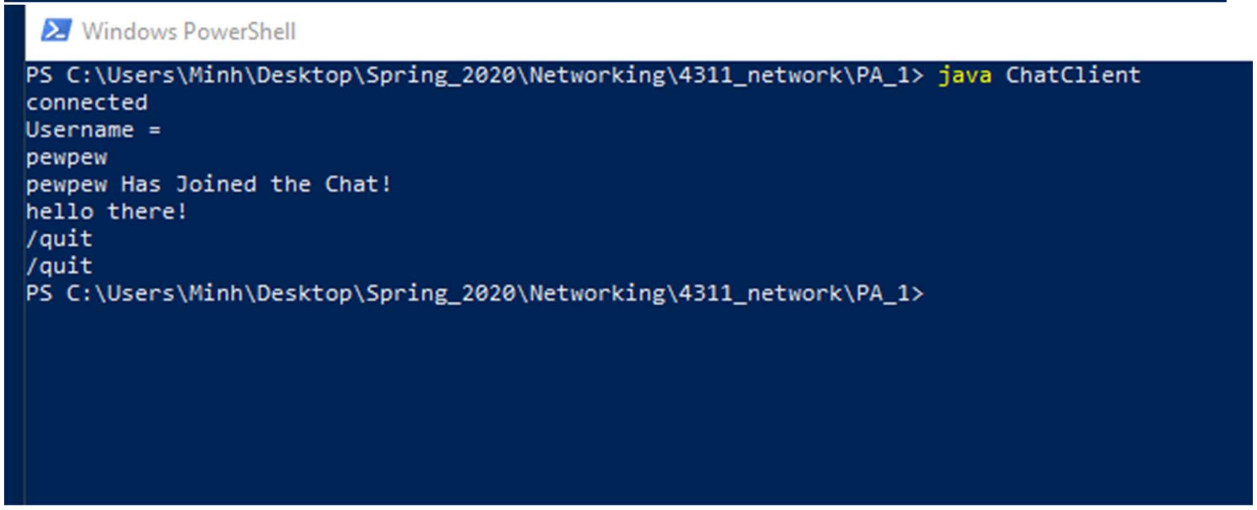
Windows PowerShell

```
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1> java ChatClient
connected
Username =
food
food Has Joined the Chat!
pewpew Has Joined the Chat!
hello there!
/quit
bye bye pewpew
```



Windows PowerShell

```
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1> java ChatClient
connected
Username =
bob
bob Has Joined the Chat!
food Has Joined the Chat!
pewpew Has Joined the Chat!
hello there!
hello there!
/quit
bye bye pewpew
```



Windows PowerShell

```
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1> java ChatClient
connected
Username =
pewpew
pewpew Has Joined the Chat!
hello there!
/quit
/quit
PS C:\Users\Minh\Desktop\Spring_2020\Networking\4311_network\PA_1>
```