Project Report

By: Team 33
Brian Pham – bnp200000
Nick Tollinger – ngt190001
Akshaan Singh – axs210087

Date: 03/21/2025

Table of Contents

Introduction

In this project, our team created a network-based application that provides a solution when a basic mathematical equation is entered (such as 23 – 11, or 34 + 4, etc.). The application is split into 2 separate parts, the server and client, where the server runs a centralized Math server that listens for client connections and carries out the arithmetic calculation requested by the client, alongside logging the join, usage, and termination of the said client. The server also allows multiple clients to connect simultaneously and carries out the request for each client in the order it receives them, while also logging their information. The client gives a name to itself for the server, and after getting the acknowledgment from the server, it is allowed to send equations to the server. The client can close the connection at any moment by inputting # in the terminal. The purpose of this project is to understand how network protocols are designed and how to properly log client data in a server; as such, constant logs are created in the server machine with requests, responses, times, etc.

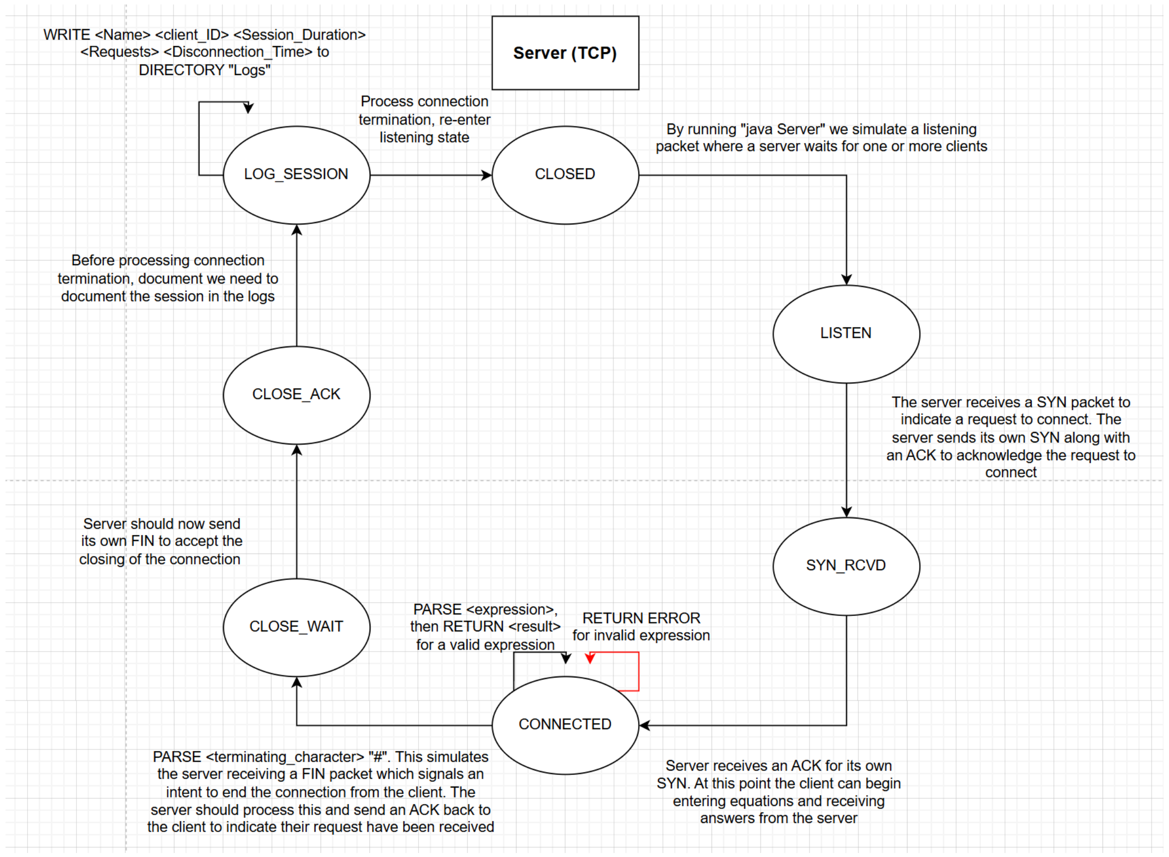
The contributions are as follows:

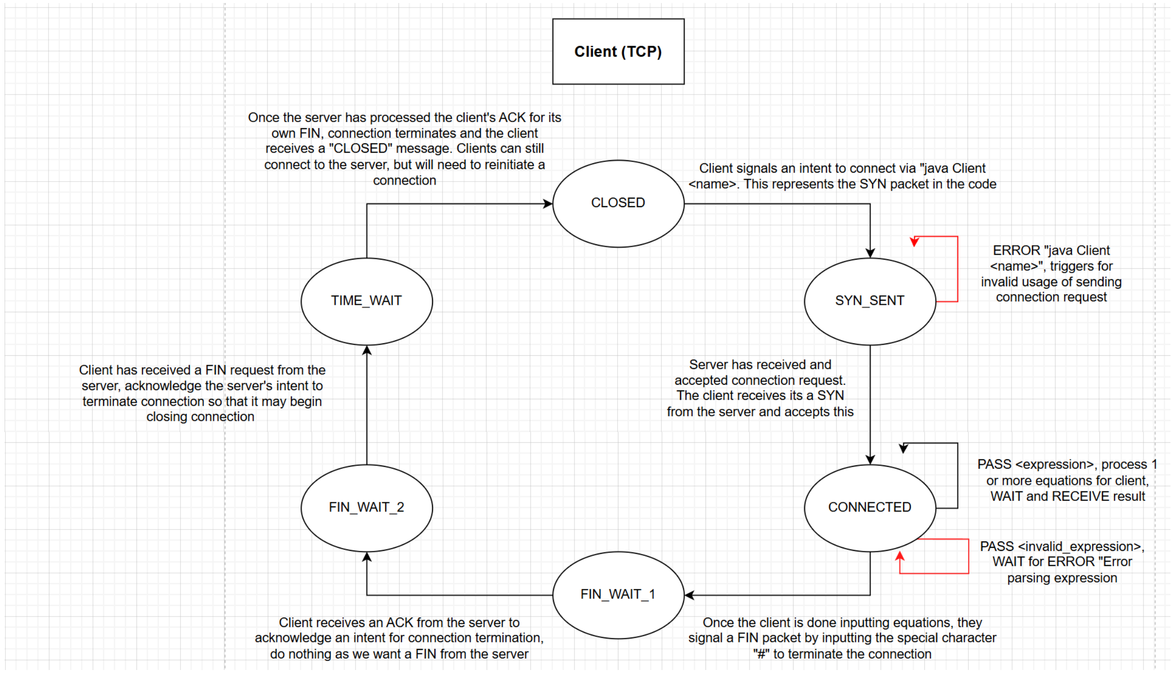
- Brian Pham: Creating, writing, compiling, and sharing the code. Wrote the documentation for the server and client, connected client middleware, and calculator backend code.
- Nick Tollinger: Creating the Protocol Design.
- Akshaan Singh: Creating and Compiling the Project Report.

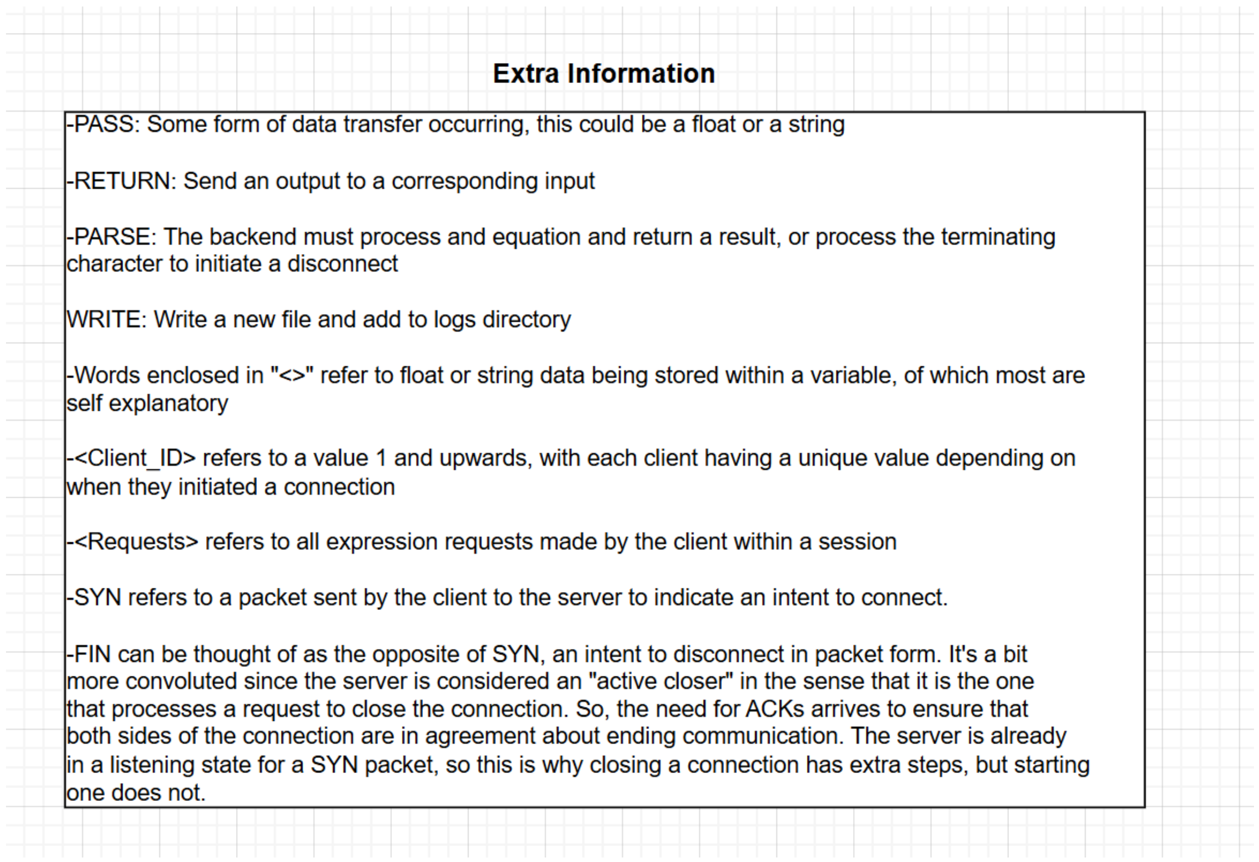
The project is mostly written in Visual Studio Code and shared on GitHub with the rest of the team for ease of access. The provided code from the instructor acts as a basis from which additional code is written to better suit the needs of the project.

To run the code, once the folder is downloaded, extract the code and open the folder in a terminal. Run the command: *javac *.java*, which will create your Java classes required for the project. Once that is done, run: *java Server*, which will run the server file and create a local server on your machine. On a different terminal opened from the same folder, run: *java Client [name]*, which will create a client of the provided name and connect to your server. The client-server architecture is now set up, and you can send your equations to the server, where it will send you the result upon request. A README file is attached, explaining the setup instructions.

Protocol Design


Server (TCP)
WRITE <Name> <client_ID> <Session_Duration> <Requests> <Disconnection_Time> to DIRECTORY "Logs"
LOG_SESSION
Process connection termination, re-enter listening state
CLOSED
By running "java Server" we simulate a listening packet where a server waits for one or more clients
LISTEN
The server receives a SYN packet to indicate a request to connect. The server sends its own SYN along with an ACK to acknowledge the request to connect
SYN_RCVD
Before processing connection termination, document we need to document the session in the logs
CLOSE_ACK
Server should now send its own FIN to accept the closing of the connection
CLOSE_WAIT
PARSE <expression>, then RETURN <result> for a valid expression
RETURN ERROR for invalid expression
CONNECTED
PARSE <terminating_character> "#". This simulates the server receiving a FIN packet which signals an intent to end the connection from the client. The server should process this and send an ACK back to the client to indicate their request have been received
Server receives an ACK for its own SYN. At this point the client can begin entering equations and receiving answers from the server


Client (TCP)
Once the server has processed the client's ACK for its own FIN, connection terminates and the client receives a "CLOSED" message. Clients can still connect to the server, but will need to reinitiate a connection
CLOSED
Client signals an intent to connect via "java Client <name>. This represents the SYN packet in the code
TIME_WAIT
SYN_SENT
ERROR "java Client <name>", triggers for invalid usage of sending connection request
Client has received a FIN request from the server, acknowledge the server's intent to terminate connection so that it may begin closing connection
Server has received and accepted connection request. The client receives its a SYN from the server and accepts this
PASS <expression>, process 1 or more equations for client, WAIT and RECEIVE result
FIN_WAIT_2
CONNECTED
PASS <invalid_expression>, WAIT for ERROR "Error parsing expression
FIN_WAIT_1
Client receives an ACK from the server to acknowledge an intent for connection termination, do nothing as we want a FIN from the server
Once the client is done inputting equations, they signal a FIN packet by inputting the special character "#" to terminate the connection



Extra Information

-PASS: Some form of data transfer occurring, this could be a float or a string

-RETURN: Send an output to a corresponding input

-PARSE: The backend must process and equation and return a result, or process the terminating character to initiate a disconnect

WRITE: Write a new file and add to logs directory

-Words enclosed in "<>" refer to float or string data being stored within a variable, of which most are self explanatory

-<Client_ID> refers to a value 1 and upwards, with each client having a unique value depending on when they initiated a connection

-<Requests> refers to all expression requests made by the client within a session

-SYN refers to a packet sent by the client to the server to indicate an intent to connect.

-FIN can be thought of as the opposite of SYN, an intent to disconnect in packet form. It's a bit more convoluted since the server is considered an "active closer" in the sense that it is the one that processes a request to close the connection. So, the need for ACKs arrives to ensure that both sides of the connection are in agreement about ending communication. The server is already in a listening state for a SYN packet, so this is why closing a connection has extra steps, but starting one does not.

The following diagram represents the protocol design for the client-server application as it involves communication between a Client, a Server, a Calculator backend, and a Logging System. The Client connects to the server by sending a PASS request that contains “java Client <Name>”, where the command has the client’s identifier. If the format is incorrect, an ERROR is returned. Upon a connection, the client sends an expression for the server to process through Port 5000, Socket 1. Once again, if the server has an issue, then it sends an ERROR back to the client. Using the PARSE request, the client sends the calculator backend the expression, and sends the computed result back to the server using the RETURN response. Using the RETURN response, the server sends the client the result. When the client sends the terminating character ‘#’ to the server, the server sends the logging system the required information using the PASS request, while the server logging system logs the information using the WRITE request. The information is written to a logs directory for future reference, and the client is disconnected from the server.

Lessons Learned

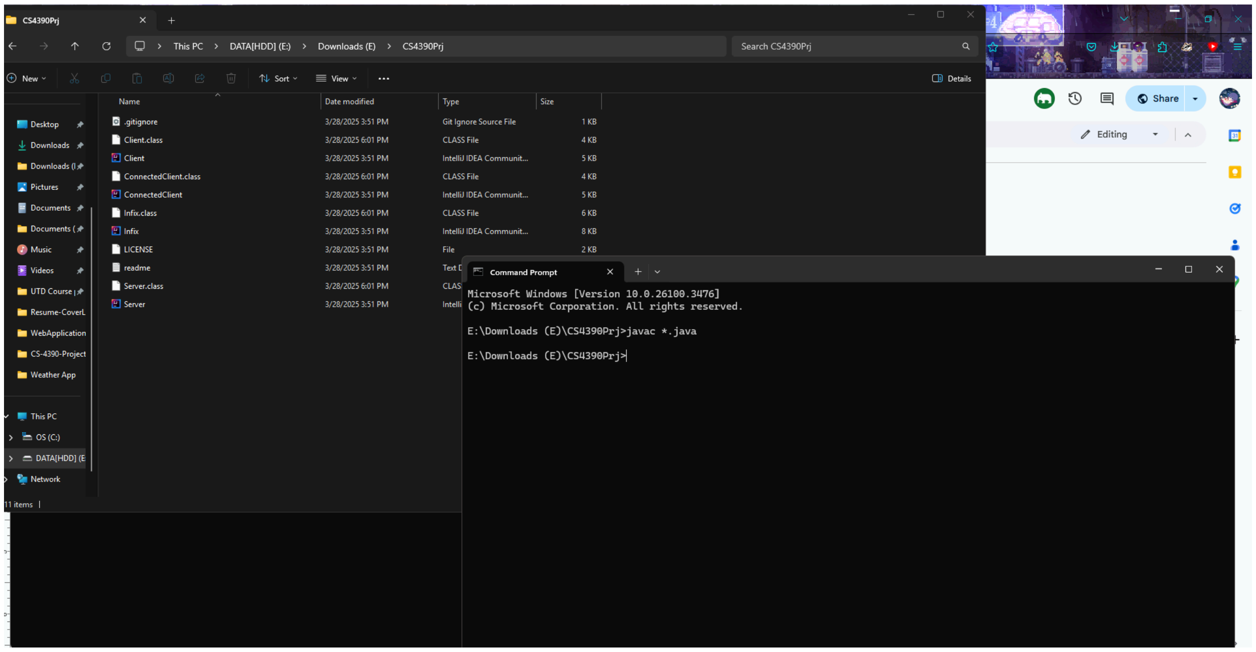
The primary objective of this project was to learn how to set up a basic client-server connection and create a personalized protocol structure for the network application. As such, the first lesson learnt was how clients and servers communicate with each other, what is recorded, what should be recorded, how, where, etc. Learning how to set up a connection between them was the first important step, and through this project, we can truly understand how these machines communicate with each other.

The next lesson learnt was how to design a protocol to fit the project's needs, as creating our own protocol allows an understanding of how they work, what they achieve, and how and why we should implement our own. Protocols, especially network protocols, are the backbone of internet security and connectivity, hence it was required to learn about them as much as possible.

Finally, the last lesson learnt was how to share information between the server and the client, and how to separate the work so that the server needs to only log the client information while another backend system computes the response for the client.

Screenshots

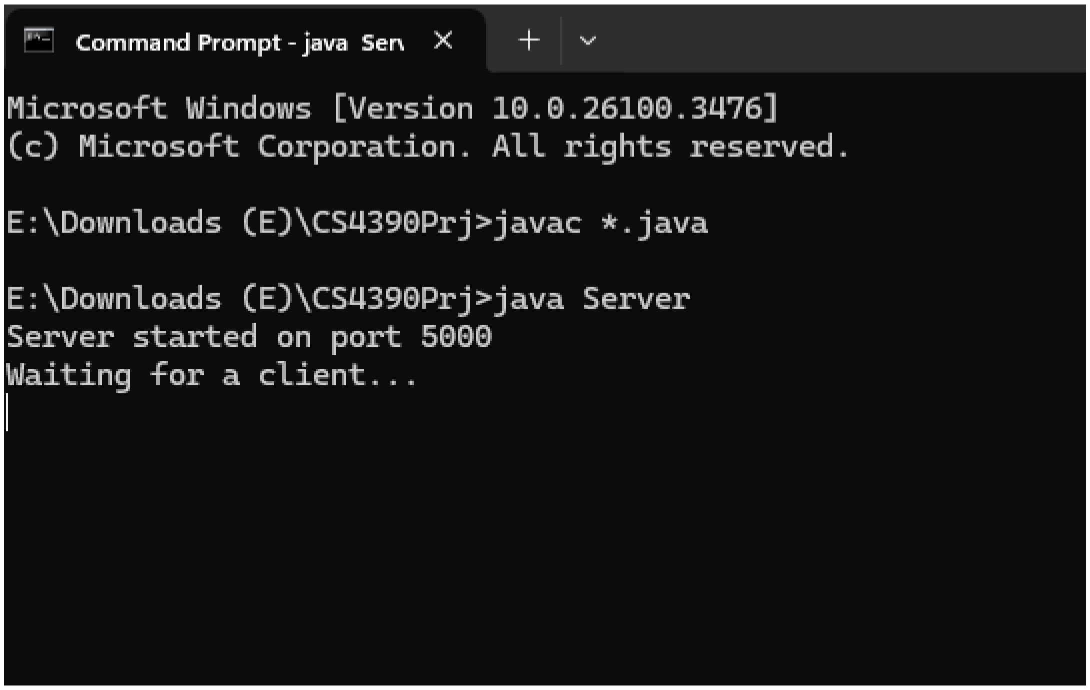
Running the initial command:



Name	Date modified	Type	Size
.gitignore	3/28/2025 3:51 PM	Git Ignore Source File	1 KB
Client.class	3/28/2025 6:01 PM	CLASS File	4 KB
Client	3/28/2025 3:51 PM	IntelliJ IDEA Communit...	5 KB
ConnectedClient.class	3/28/2025 6:01 PM	CLASS File	4 KB
ConnectedClient	3/28/2025 3:51 PM	IntelliJ IDEA Communit...	5 KB
Infix.class	3/28/2025 6:01 PM	CLASS File	6 KB
Infix	3/28/2025 3:51 PM	IntelliJ IDEA Communit...	8 KB
LICENSE	3/28/2025 3:51 PM	File	2 KB
readme	3/28/2025 3:51 PM	Text D	
Server.class	3/28/2025 6:01 PM	CLAS	
Server	3/28/2025 3:51 PM	Intelli	

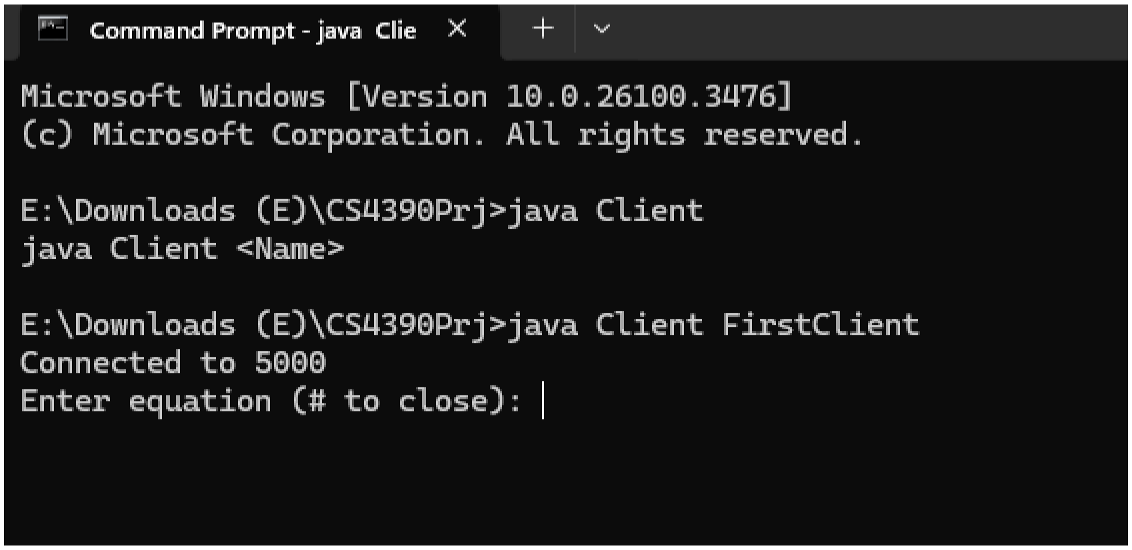
```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>javac *.java

E:\Downloads (E)\CS4390Prj>
```

Running the Server command:

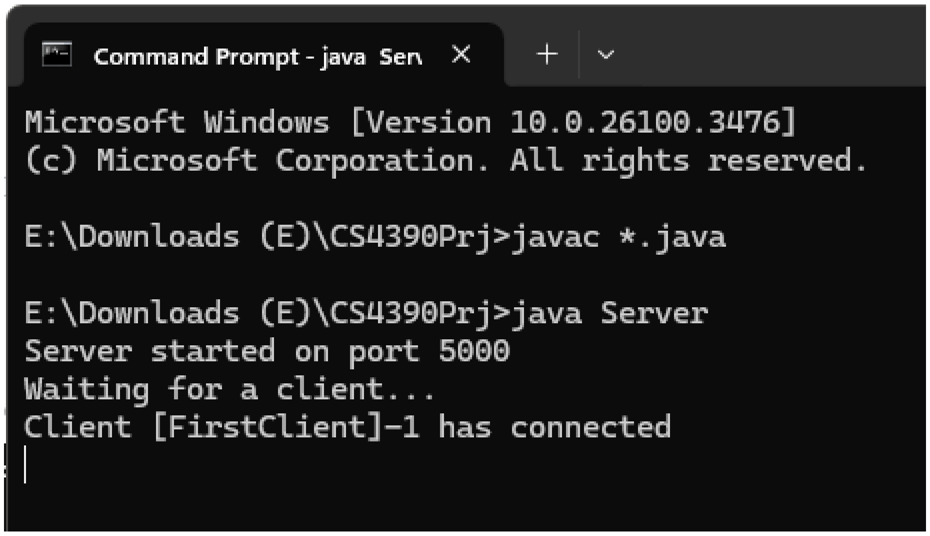


```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>javac *.java

E:\Downloads (E)\CS4390Prj>java Server
Server started on port 5000
Waiting for a client...
```

Running the Client command in a new terminal:

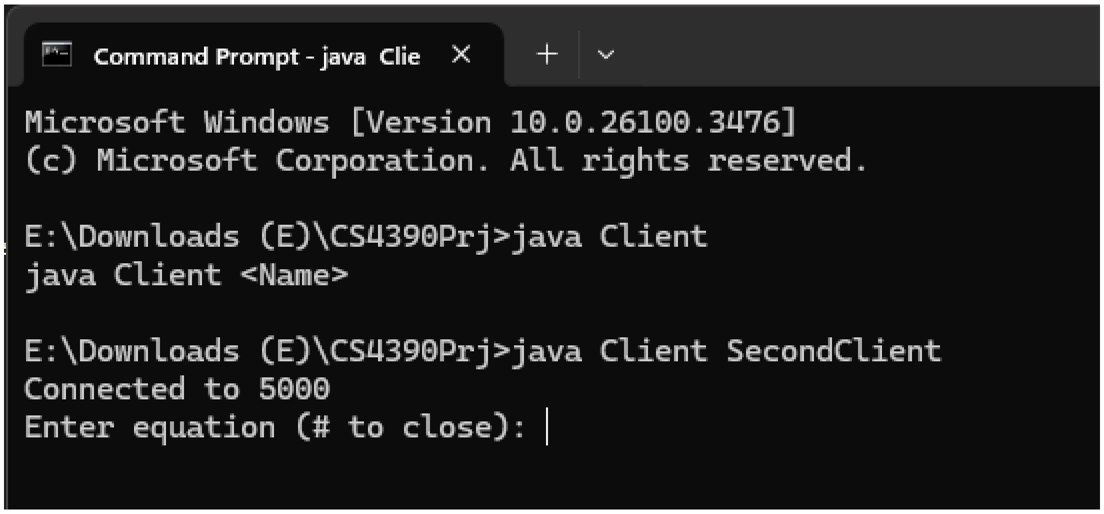


```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>java Client
java Client <Name>

E:\Downloads (E)\CS4390Prj>java Client FirstClient
Connected to 5000
Enter equation (# to close):
```



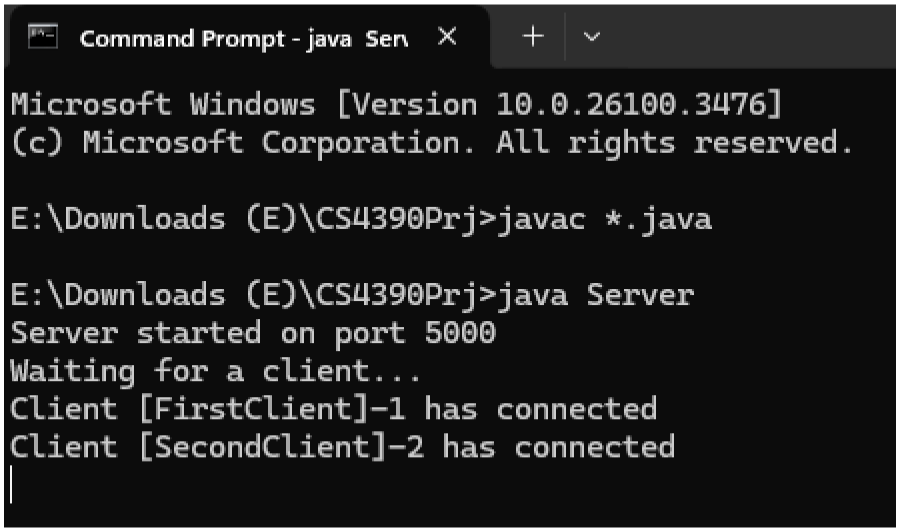
```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>javac *.java

E:\Downloads (E)\CS4390Prj>java Server
Server started on port 5000
Waiting for a client...
Client [FirstClient]-1 has connected
```

Connecting the second client:



Command Prompt - java Clie

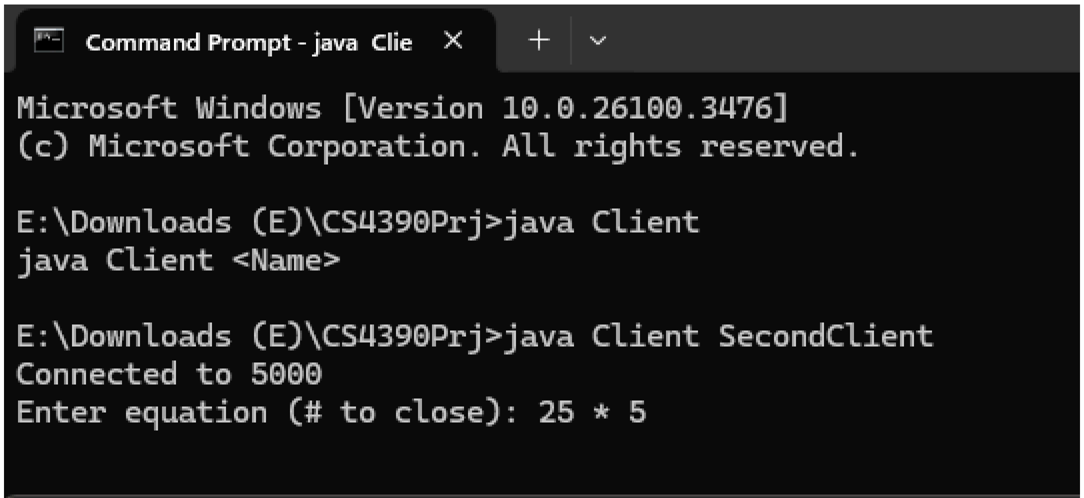
```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>java Client
java Client <Name>

E:\Downloads (E)\CS4390Prj>java Client SecondClient
Connected to 5000
Enter equation (# to close):
```



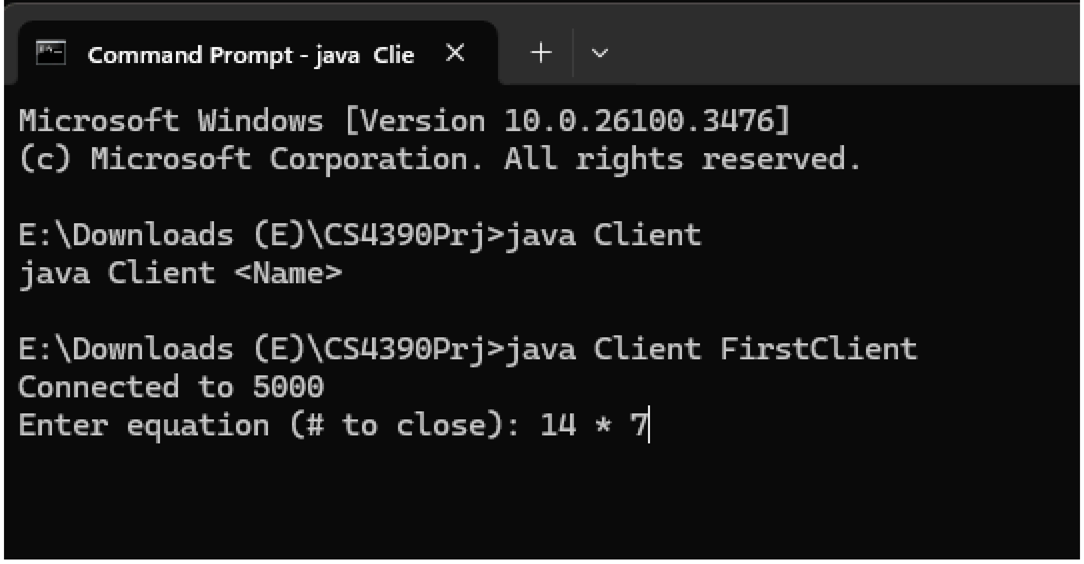
Command Prompt - java Serv

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>javac *.java

E:\Downloads (E)\CS4390Prj>java Server
Server started on port 5000
Waiting for a client...
Client [FirstClient]-1 has connected
Client [SecondClient]-2 has connected
```

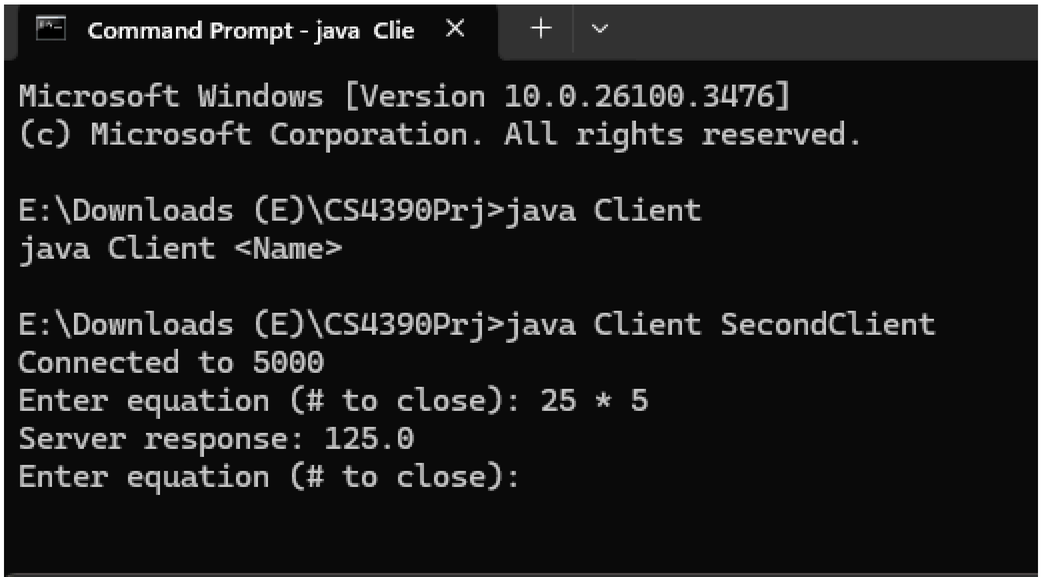
Sending an equation from both clients:



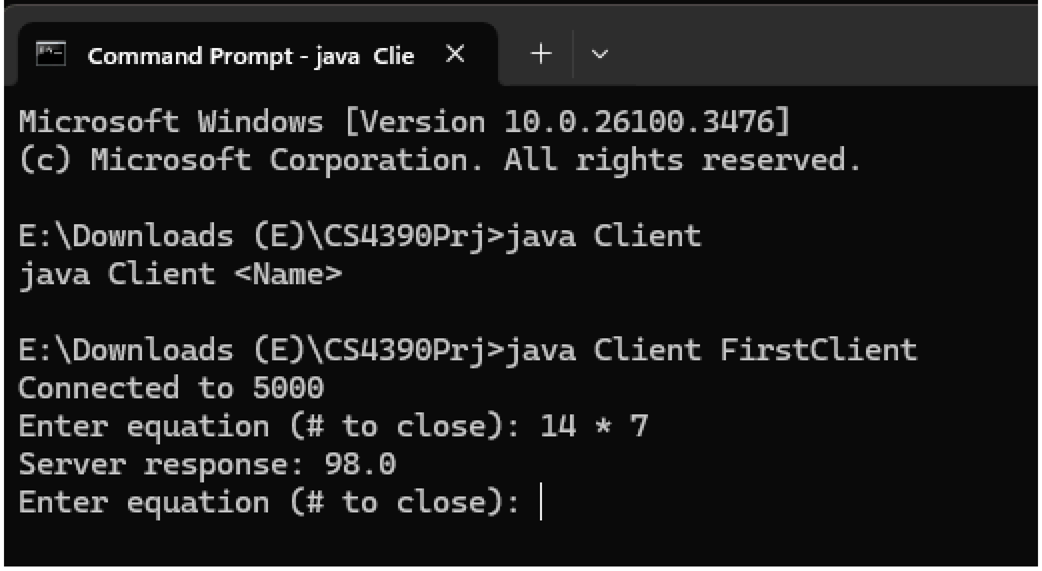
```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>java Client
java Client <Name>

E:\Downloads (E)\CS4390Prj>java Client SecondClient
Connected to 5000
Enter equation (# to close): 25 * 5
```



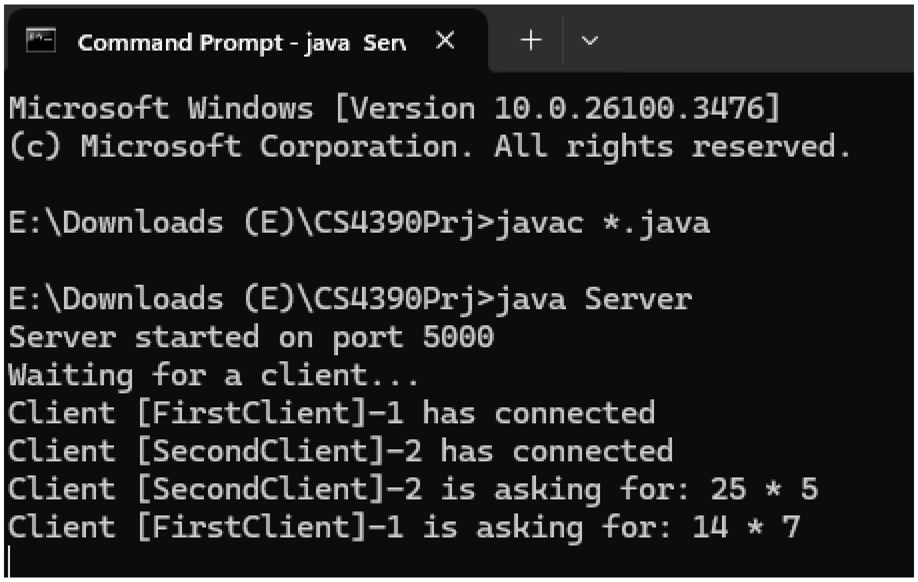
```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>java Client
java Client <Name>

E:\Downloads (E)\CS4390Prj>java Client FirstClient
Connected to 5000
Enter equation (# to close): 14 * 7
```



```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>java Client
java Client <Name>

E:\Downloads (E)\CS4390Prj>java Client SecondClient
Connected to 5000
Enter equation (# to close): 25 * 5
Server response: 125.0
Enter equation (# to close):
```

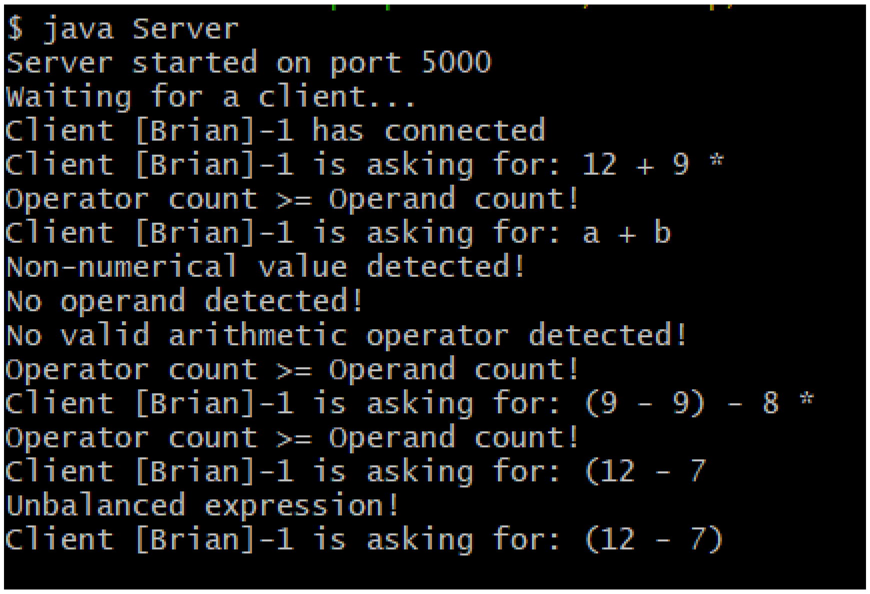


```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>java Client
java Client <Name>

E:\Downloads (E)\CS4390Prj>java Client FirstClient
Connected to 5000
Enter equation (# to close): 14 * 7
Server response: 98.0
Enter equation (# to close):
```

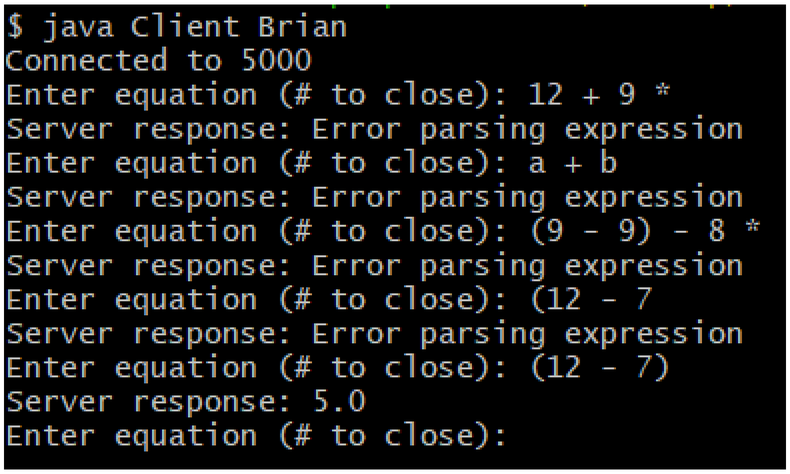


```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>javac *.java

E:\Downloads (E)\CS4390Prj>java Server
Server started on port 5000
Waiting for a client...
Client [FirstClient]-1 has connected
Client [SecondClient]-2 has connected
Client [SecondClient]-2 is asking for: 25 * 5
Client [FirstClient]-1 is asking for: 14 * 7
```

Sending malformed input

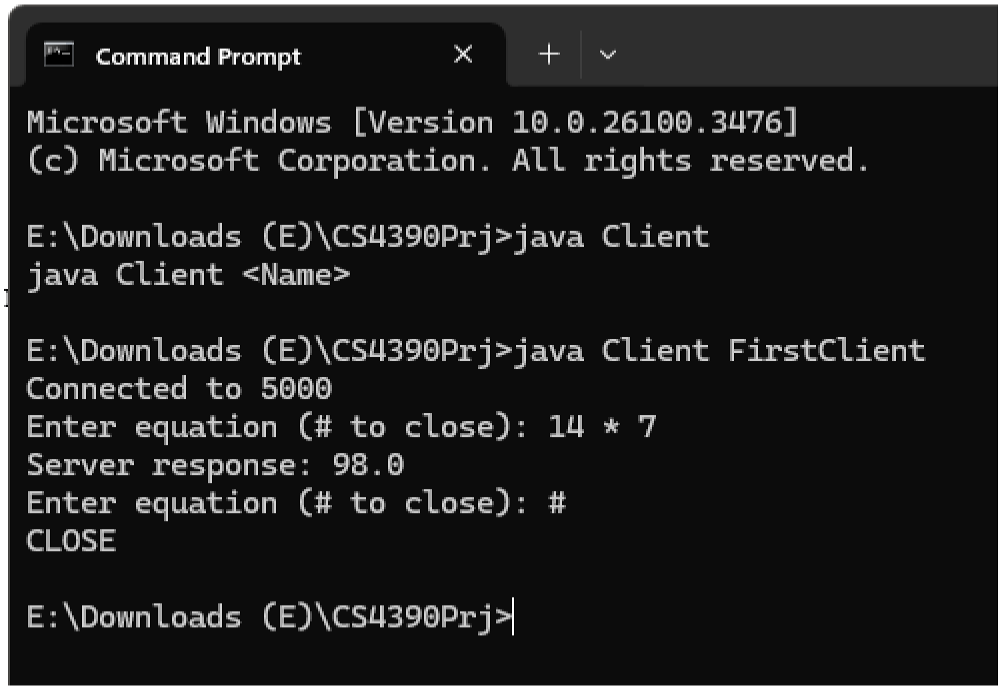


```
$ java Server
Server started on port 5000
Waiting for a client...
Client [Brian]-1 has connected
Client [Brian]-1 is asking for: 12 + 9 *
Operator count >= Operand count!
Client [Brian]-1 is asking for: a + b
Non-numerical value detected!
No operand detected!
No valid arithmetic operator detected!
Operator count >= Operand count!
Client [Brian]-1 is asking for: (9 - 9) - 8 *
Operator count >= Operand count!
Client [Brian]-1 is asking for: (12 - 7
Unbalanced expression!
Client [Brian]-1 is asking for: (12 - 7)
```

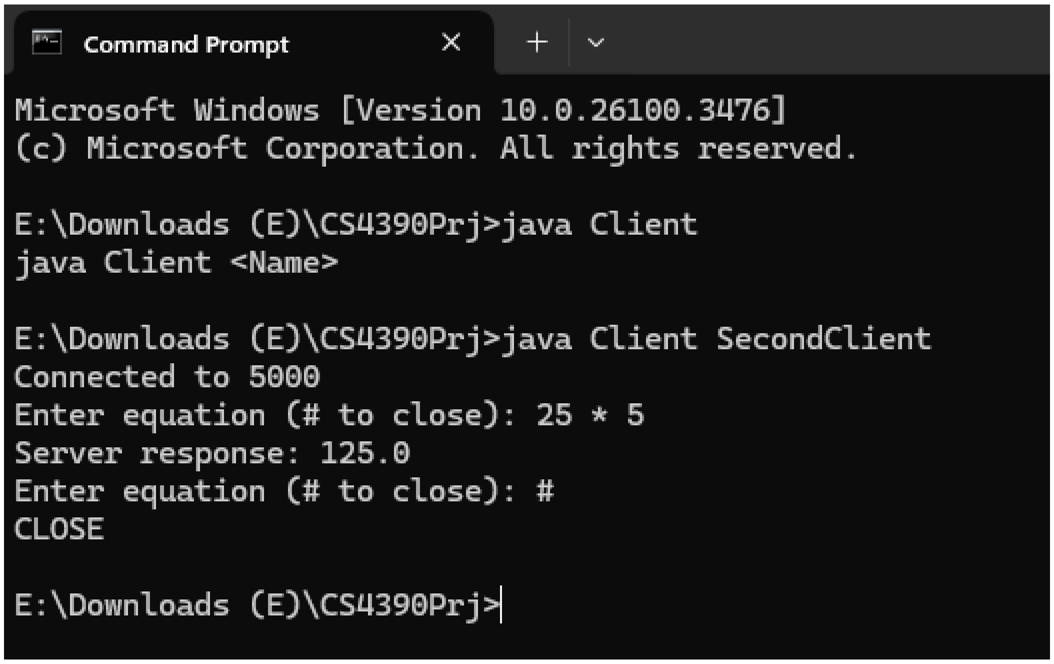


```
$ java Client Brian
Connected to 5000
Enter equation (# to close): 12 + 9 *
Server response: Error parsing expression
Enter equation (# to close): a + b
Server response: Error parsing expression
Enter equation (# to close): (9 - 9) - 8 *
Server response: Error parsing expression
Enter equation (# to close): (12 - 7
Server response: Error parsing expression
Enter equation (# to close): (12 - 7)
Server response: 5.0
Enter equation (# to close):
```

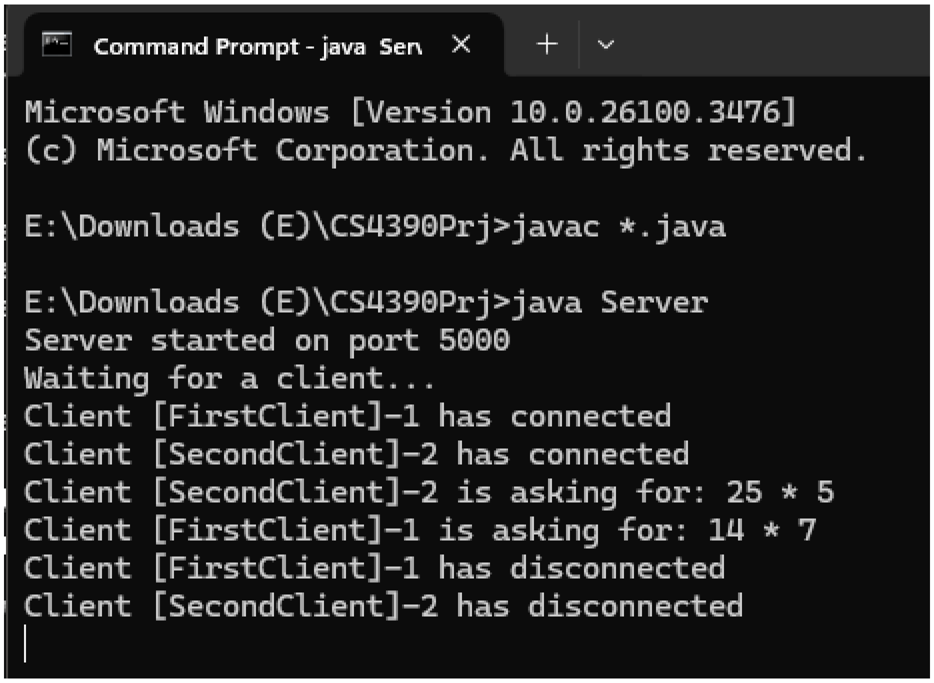
Closing the connection:



```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>java Client
java Client <Name>

E:\Downloads (E)\CS4390Prj>java Client FirstClient
Connected to 5000
Enter equation (# to close): 14 * 7
Server response: 98.0
Enter equation (# to close): #
CLOSE

E:\Downloads (E)\CS4390Prj>
```



```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>java Client
java Client <Name>

E:\Downloads (E)\CS4390Prj>java Client SecondClient
Connected to 5000
Enter equation (# to close): 25 * 5
Server response: 125.0
Enter equation (# to close): #
CLOSE

E:\Downloads (E)\CS4390Prj>
```


Command Prompt - java Serv

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

E:\Downloads (E)\CS4390Prj>javac *.java

E:\Downloads (E)\CS4390Prj>java Server
Server started on port 5000
Waiting for a client...
Client [FirstClient]-1 has connected
Client [SecondClient]-2 has connected
Client [SecondClient]-2 is asking for: 25 * 5
Client [FirstClient]-1 is asking for: 14 * 7
Client [FirstClient]-1 has disconnected
Client [SecondClient]-2 has disconnected
```