CS2003 Practical 1 Report

Tutor: Dr. Tristan Henderson

Matriculation ID: 230030861

2023-09-24

Overview

In Practical 1, our assignment was to develop 2 programs facilitating a simple client-server interaction. This interaction operates on a text-based protocol where communication occurs through a sequence of messages. Both the client and server use TCP for communication and send ASCII strings as data. I successfully achieved this goal. Following the provided instructions, my server waits for TCP connections, and responds with the appropriate strings. Similarly, my client connects to the server, and initiates the protocol. Both of these programs adhere closely to the given guidelines.

Design

For my design of SimpleClient, I used an enumeration, "ClientState," representing the three states of the client during the interaction (Greeting, Advising, and Closing) (Oracle). Each state has associated phrases and defines expected server responses. The "main" method accepts two command-line arguments, the server IP address and port number. The client creates a socket for communication with the server, and it initializes input and output streams for sending and receiving messages (Allison, Colin). The client enters a loop where it sends and receives messages to and from the server based on the associated state until the server responds with "YOU'RE WELCOME." The client then disconnects. If an error occurs throughout the program, the terminal will print out why (Bhatti, Saleem September 2019).

Similarly to SimpleClient, I used an enumeration, "ServerState," representing the four states of the server during the interaction (Greeting, Advise, Close, and Restart) (Oracle). Each state defines the expected message from the client and the server's response. The "main" method accepts one argument, the port number. The server creates a "ServerSocket" to wait and listen for client connections, and it also initializes input and output streams for communication with the client (Henderson, Tristan 2021). The server enters a loop where it listens for client connections, handles each client connection, and processes messages based on the state. If a client sends an incorrect message, the server responds with an error message and remains in the same state until the correct

message is received. After completing the interaction, the server returns to the greeting state and remains open, waiting for more connections. If an error occurs throughout the program, the terminal will print out why (Bhatti, Saleem September 2019).

The use of enumerations for states makes the code more organized, easier to read, and more maintainable. The command-line arguments allow for more adaptability when connecting from different IP addresses. The sockets adhere to the standard model. The loop-based design allows for continuous interaction, and the server's ability to remain open and handle multiple clients makes it a more realistic application. The error handling helps with debugging and usability.

Testing

What is being tested?	Pre-conditions	Expected Outcome	Actual Outcome
Basic Interaction	The user runs the programs with arguments for the IP address and port for the client and port for the server.	Server: Waiting for connection on port: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8086]... Client connected from /138.251.22.77 Client says: HELLO ADVISER Client says: ADVISE ME ON TO CS2003 Client says: THANK YOU Client disconnected Client: Server says: HELLO ADVISEE Server says: YOU ARE ADVISED ON TO CS2003 Server says: YOU'RE WELCOME	Server: Waiting for connection on port: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8086]... Client connected from /138.251.22.77 Client says: HELLO ADVISER Client says: ADVISE ME ON TO CS2003 Client says: THANK YOU Client disconnected Client: Server says: HELLO ADVISEE Server says: YOU ARE ADVISED ON TO CS2003 Server says: YOU'RE WELCOME **See Figure 1a and 1b**
Incorrect Client Message	The user inputs an incorrect message to the server.	Server: Client connected from /138.251.22.77 Client says: h Client says: HELLO ADVISER Client says: h Client says: ADVISE ME ON TO CS2003 Client says: h Client says: THANK YOU Client disconnected Client: h Incorrect input... Expecting: HELLO ADVISER HELLO ADVISER HELLO ADVISEE h Incorrect input... Expecting: ADVISE ME ON TO CS2003 ADVISE ME ON TO CS2003 YOU ARE ADVISED ON TO CS2003	Server: Client connected from /138.251.22.77 Client says: h Client says: HELLO ADVISER Client says: h Client says: ADVISE ME ON TO CS2003 Client says: h Client says: THANK YOU Client disconnected Client: h Incorrect input... Expecting: HELLO ADVISER HELLO ADVISER HELLO ADVISEE h Incorrect input... Expecting: ADVISE ME ON TO CS2003 ADVISE ME ON TO CS2003 YOU ARE ADVISED ON TO CS2003

		h Incorrect input... Expecting: THANK YOU THANK YOU YOU'RE WELCOME	h Incorrect input... Expecting: THANK YOU THANK YOU YOU'RE WELCOME **See Figure 2a and 2b**
Multiple Connections	Two different clients connect to the same server.	Server: Client connected from /138.251.22.77 Client says: HELLO ADVISER Client says: ADVISE ME ON TO CS2003 Client says: THANK YOU Client disconnected Client connected from /138.251.22.77 Client says: HELLO ADVISER Client says: ADVISE ME ON TO CS2003 Client says: THANK YOU Client disconnected Client 1: java SimpleClient 138.251.22.77 8086 Server says: HELLO ADVISEE Server says: YOU ARE ADVISED ON TO CS2003 Server says: YOU'RE WELCOME Client 2: java SimpleClient 138.251.22.77 8086 Server says: HELLO ADVISEE Server says: YOU ARE ADVISED ON TO CS2003 Server says: YOU'RE WELCOME	Server: Client connected from /138.251.22.77 Client says: HELLO ADVISER Client says: ADVISE ME ON TO CS2003 Client says: THANK YOU Client disconnected Client connected from /138.251.22.77 Client says: HELLO ADVISER Client says: ADVISE ME ON TO CS2003 Client says: THANK YOU Client disconnected Client 1: java SimpleClient 138.251.22.77 8086 Server says: HELLO ADVISEE Server says: YOU ARE ADVISED ON TO CS2003 Server says: YOU'RE WELCOME Client 2: java SimpleClient 138.251.22.77 8086 Server says: HELLO ADVISEE Server says: YOU ARE ADVISED ON TO CS2003 Server says: YOU'RE WELCOME **See Figure 3a, 3b, and 3c**
Different Server Locations	The server is ran on a different teaching server than Lyrane	Server: java SimpleServer 8086 Waiting for connection on port: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8086]... Client connected from /138.251.22.78 Client says: HELLO ADVISER Client says: ADVISE ME ON TO CS2003 Client says: THANK YOU Client disconnected Client: java SimpleClient 138.251.22.78 8086 Server says: HELLO ADVISEE Server says: YOU ARE ADVISED ON TO CS2003 Server says: YOU'RE WELCOME	Server: java SimpleServer 8086 Waiting for connection on port: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8086]... Client connected from /138.251.22.78 Client says: HELLO ADVISER Client says: ADVISE ME ON TO CS2003 Client says: THANK YOU Client disconnected Client: java SimpleClient 138.251.22.78 8086 Server says: HELLO ADVISEE Server says: YOU ARE ADVISED ON TO CS2003 Server says: YOU'RE WELCOME **See Figure 4a and 4b**
Client Begins First	The client begins with connection	Client: java SimpleClient 138.251.22.77 8086	Client: java SimpleClient 138.251.22.77 8086

	before the server.	Error: Connection refused	Error: Connection refused **See Figure 5**
Incorrect Server Argument	1. The server receives a non-number for the port 2. The server attempts to use a port already in use. 3. Incorrect number of arguments	1. Server: java SimpleServer 808b Port must be a number 2. Server: java SimpleServer 80 Error: Permission denied 3. Server: java SimpleServer 8086 g Usage: java SimpleServer <port>	1. Server: java SimpleServer 808b Port must be a number 2. Server: java SimpleServer 80 Error: Permission denied 3. Server: java SimpleServer 8086 g Usage: java SimpleServer <port> **See Figure 6a, 6b, and 6c**
Incorrect Client Arguments	1. The client receives a non-number for the port 2. The client attempts to use an incorrect IP address 3. Incorrect number of arguments	1. Client: java SimpleClient 138.251.22.77 808b Port must be a number 2. Client: java SimpleClient 138.251.22.87 8086 Error: No route to host 3. Client: java SimpleClient 138.251.22.77 8086 g Usage: java SimpleClient <IPaddress> <port>	1. Client: java SimpleClient 138.251.22.77 808b Port must be a number 2. Client: java SimpleClient 138.251.22.87 8086 Error: No route to host 3. Client: java SimpleClient 138.251.22.77 8086 g Usage: java SimpleClient <IPaddress> <port> **See Figure 7a, 7b, and 7c**

Figure 1:

```
Waiting for connection on port: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8086]...
Client connected from /138.251.22.77
Client says: HELLO ADVISER
Client says: ADVISE ME ON TO CS2003
Client says: THANK YOU
Client disconnected
```

Figure 1a

```
Server says: HELLO ADVISEE
Server says: YOU ARE ADVISED ON TO CS2003
Server says: YOU'RE WELCOME
```

Figure 1b

Figure 2:

```
Client connected from /138.251.22.77
Client says: h
Client says: HELLO ADVISER
Client says: h
Client says: ADVISE ME ON TO CS2003
Client says: h
Client says: THANK YOU
Client disconnected
```

Figure 2a

```
tvt1@lyrane:~/Documents/CS2003/P1-remote-server $ nc 138.251.22.77 8086
h
Incorrect input... Expecting: HELLO ADVISER
HELLO ADVISER
HELLO ADVISEE
h
Incorrect input... Expecting: ADVISE ME ON TO CS2003
ADVISE ME ON TO CS2003
YOU ARE ADVISED ON TO CS2003
h
Incorrect input... Expecting: THANK YOU
THANK YOU
YOU'RE WELCOME
```

Figure 2b

Figure 3:

```
Client connected from /138.251.22.77
Client says: HELLO ADVISER
Client says: ADVISE ME ON TO CS2003
Client says: THANK YOU
Client disconnected
Client connected from /138.251.22.77
Client says: HELLO ADVISER
Client says: ADVISE ME ON TO CS2003
Client says: THANK YOU
Client disconnected
```

Figure 3a

```
tvt1@lyrane:~/Documents/CS2003/P1-remote-server $ java SimpleClient 138.251.22.77 8086
Server says: HELLO ADVISEE
Server says: YOU ARE ADVISED ON TO CS2003
Server says: YOU'RE WELCOME
```

Figure 3b

```
tvt1@lyrane:~/Documents/CS2003/P1-remote-server $ java SimpleClient 138.251.22.77 8086
Server says: HELLO ADVISEE
Server says: YOU ARE ADVISED ON TO CS2003
Server says: YOU'RE WELCOME
```

Figure 3c

Figure 4

```
tvt1@klovia:~/Documents/CS2003/P1-remote-server $ java SimpleServer 8086
Waiting for connection on port: ServerSocket[addr=0.0.0.0/0.0.0.0,localport=8086]...
Client connected from /138.251.22.78
Client says: HELLO ADVISER
Client says: ADVISE ME ON TO CS2003
Client says: THANK YOU
Client disconnected
```

Figure 4a

```
tvt1@klovia:~/Documents/CS2003/P1-remote-server $ java SimpleClient 138.251.22.78 808
Server says: HELLO ADVISEE
Server says: YOU ARE ADVISED ON TO CS2003
Server says: YOU'RE WELCOME
```

Figure 4b

Figure 5

```
tvt1@lyrane:~/Documents/CS2003/P1-remote-server $ java SimpleClient 138.251.22.77 8086
Error: Connection refused
```

Figure 6

```
^Ctvt1@lyrane:~/Documents/CS2003/P1-remote-server $ java SimpleServer 808b
Port must be a number
```

Figure 6a

```
tvt1@lyrane:~/Documents/CS2003/P1-remote-server $ java SimpleServer 80
Error: Permission denied
```

Figure 6b

```
^Ctvt1@lyrane:~/Documents/CS2003/P1-remote-server $ java SimpleServer 8086 g
Usage: java SimpleServer <port>
```

Figure 6c

Figure 7

```
tvt1@lyrane:~/Documents/CS2003/P1-remote-server $ java SimpleClient 138.251.22.77 808b
Port must be a number
```

Figure 7a

```
tvt1@lyrane:~/Documents/CS2003/P1-remote-server $ java SimpleClient 138.251.22.87 8086
Error: No route to host
```

Figure 7b

```
tvt1@lyrane:~/Documents/CS2003/P1-remote-server $ java SimpleClient 138.251.22.77 8086 g
Usage: java SimpleClient <IPaddress> <port>
```

Figure 7c

Evaluation

Both of my programs should exceed expectations compared to what I was asked to do. My programs function as intended and fulfill all requirements. They handle errors, unexpected inputs, and multiple client connections. The code is easy to understand and well-formatted. I also chose to make it a FSM, which was not required. However, it could handle more complex states, and based on my enumerations, that would be easy to code.

Conclusion

I have fully completed the practical to the best of my ability. Learning how to code the client-server interaction was quite difficult, but the lab

helped it all make sense. If I had more time, I would think of more test cases to run or try to add more complexity in the states and/or inputs.

Bibliography

Oracle. "Enums (The Java™ Tutorials > Learning the Java Language > Classes and Objects)." Oracle Java Tutorials, https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html.

Allison, Colin. "DayTimeClient."

Bhatti, Saleem. "TcpClientSimpleNB." September 2019.

Henderson, Tristan. "BufferedReaderServer." 2021.