Kiet Nguyen, #0273533519

Thai Tran, #026819986

CECS 326 Sec 02

Project 1 Report

1) The program implements a client-server communication protocol using sockets. The client program (QuoteClient) connects to the server on the same port number, reads the quote message sent by the server, and prints it to the console. Inside the main method, a try-catch block is used to handle any exceptions. We connect to the server on port 6017, and the address 127.0.0.1 (standard address for loopback traffic). Then, read the quote sent by the server using a buffered reader on the input stream. Finally, it uses readline() to read the quote from buffered reader and display the quote.

   The QuoteServer program creates a server socket and binds it to port 6017. It waits for incoming connections in a continuous loop. When a client connects, a new socket object is created. It then creates a PrintWriter object that writes the quote to the output stream of the socket object. After sending the quote to the client, the socket connection is closed, and it displays the run time. If an error occurs while waiting for incoming connections, the error message is printed to the console.

2) Similarly, the second program also implements a client-server communication protocol through utilizing sockets. Once both of the client (EchoClient) and server (EchoServer) connect with each other on the same port number, the client is then able to type in anything they want and have it 'echoed' (printed) back in the terminal. We import the same classes to utilize Java API as the first program. Using the port number 6007 and the address of 127.0.0.1, both the client and server were able to successfully connect with each other and communicate through using buffered input/output streams as well as the reader.

   Specifically, the EchoServer is called first and proceeds to wait for the client to connect. Once it has found a client with the same port number, the server will then move on to accept the connection. From there, we run a continuous loop in the EchoClient to have the client input anything they want to have echoed back using a buffered reader and writing it back to the server with an input/output stream. The client is able to end the stream and connection by typing in the phrase 'Done'. From the InputStream class on the server side, the .read() method will return a -1 when the connection has been closed, allowing us to break out of the loop and end the program.
   Link to recording demo: https://youtu.be/6Fq6DjKEPYA