Echo Server Implementation

This assignment demonstrates the way to use Java programming language to create a multi-threaded Echo server. Incoming client connections are detected by the Echo Server on port 8081. Each client sends a single line of text, which the Echo Server then echoes back before cutting the connection. Until the server is manually shut off, it stays up and keeps making new connections. This project adheres to the course's learning objectives by utilizing fundamental ideas from concurrency and socket programming**.**

Code Screenshots

A screen shot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Using java.net.ServerSocket class, the Echo Server opens a listening socket on port 8081. The server uses the accept () method for accepting a client connection and launches a new thread to manage communication. Using a buffered reader, each thread receives one line of information from the client, uses a buffered writer to transmit the same message back, and then disconnects. The server can manage several clients at once without experiencing any blocking thanks to this multi-threaded technique. The implementation demonstrates how to effectively leverage Java's threading and networking APIs to create a server application that is both scalable and responsive.

This assignment reinforced my understanding of socket programming and multi-threaded server design. I successfully implemented and tested a Java Echo Server capable of handling multiple clients concurrently.

**7. References**

Use APA format. Example:

Deitel, P. J., & Deitel, H. M. (2017). *Java: How to Program* (10th ed.). Pearson.

Oracle. (n.d.). *Class ServerSocket*. Retrieved September 21, 2025, from <https://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html>