This project implements a fully-functional multi-client chat system in Java, built entirely from low-level socket programming and custom threading logic. The server uses a multi-threaded acceptor model, where each incoming client connection is handled by a dedicated ServerBroadcast instance. This instance manages two internal threads — one for receiving client messages and one for sending broadcast messages — enabling real-time, bidirectional chat communication.

The server maintains a shared list of active client output streams, allowing messages from any client (or the server console) to be broadcast to all connected clients. Each accepted socket connection is processed independently, ensuring simultaneous message handling without blocking other users.

Features:

- Dynamic thread creation for handling new client connections
- Server-side broadcasting to all clients via shared output stream lists
- Per-client receive/send threads using blocking I/O (readUTF, writeUTF)
- Graceful disconnect detection (via exceptions or "over" command)
- Automatic removal of disconnected clients from the broadcast pool
- Optional server shutdown when all clients have exited
- Console-driven server messaging for live server-to-client commands
- Multi-client support via ServerSocket.accept() running in concurrent threads

This project demonstrates deep understanding of:

- Java networking fundamentals (ServerSocket, Socket, streams)
- Thread synchronization and concurrency
- Real-time broadcast messaging
- Managing shared resources across threads
- Clean shutdown logic and exception-based lifecycle control

- Low-level blocking I/O behavior and thread coordination