Rahat Ahmed

rxa121930

CS 4348 Project 3 Summary

The purpose of this project was to simulate a movie ticket reserving system by writing a client and a server. For the protocol I decided to send simple ASCII messages, where each line was a separate message, and each message had a command or reply name and optional parameters. Most of the protocol commands and replies only needed one line; only the list of movies needed multiple lines, and I handled that with a terminating reply indicating the end of the list (however in hindsight it maybe have been easier just to send everything in one line).

Neither the server nor client were difficult to implement. Half of the server code was trivial error handling and reading the movies.txt file. Setting up the ServerSocket and client handling threads was extremely easy, and reading client commands and returning server replies wasn’t a problem either. As far as concurrent access goes, the only shared data that could possibly change was the seat count, so I made a method called reserveSeats(index, num) to use when accessing the seats array. I use it to remove from the amount of available seats, and to add back to the available seats if the user cancels his reservation by sending a negative value for the num parameter. It’s the only method I need for accessing the array, so I made reserveSeats synchronized so only one thread can use it at a time. However, I didn’t get a chance to test if this actually prevents concurrency errors, because it was difficult to do so.

Again, half of the client code was about handling trivial things like errors and incorrect user inputs. I wrote some convenience functions to ask the user for input, but otherwise the client implementation was as straightforward as the server’s.

Overall, I feel that this project was the easiest one of all the projects in the course. Java handles so much for you when dealing with Sockets, there’s nothing to really cause trouble when developing with them.