This is a multithreaded chatroom application that was written in the Java programming language. The involved classes are as follows:

Server
Client (this class will have a separate thread strictly for reading)
User
SessionThread (this class will be associated with multiple threads)

Rather than having the User and SessionThread classes as subclasses in the Server class, they were left to be on their own for better code organization.

On the Server side, the only thing that is required is to load the program in the usual fashion, i.e. type “java Server” in terminal. It will display when it is: accepting an incoming connection, when a user joins, and when a user leaves.

The chatroom chat history will always be loaded whenever the Server is ran via serialization, which will load it from an ArrayList of type String. The only time that the chatroom history gets saved is in the SessionThread class after one of the running threads receives a message from the user (commands do not get saved).

The users will be able to differentiate themselves from one another not just by name, but also by a color scheme (thank you Professor Russell)--and they will always be different colors from one another.

The commands are as follows:

@name -- only used once the Client program is ran, to give the Server a name to create the user.
@private -- used to private message another active user.
@end -- used to end the private messaging with a user.
@who -- used to list the active users and active users who are being targeted for private messaging. NOTE: Not sure why we had to differentiate between them. Don’t think this would ever happen in the real world.
@exit -- used to exit the chatroom.

EXTRA CREDIT:

If a user attempts to private message another user who is already receiving private messages from a different user, then that user can retry to initiate a private message within a timespan of ~10 seconds (the user will be made aware of this and a countdown of messages will follow).

P.S. I may restructure my code in the future for a more robust application--maybe even make a GUI for it. All in all, it was a fun assignment.