The design is implemented through several utility and main classes, which are as follows:

On the Client side, there are three classes which control operation:

Client.java – This is the main class on the client side, it connects to the server and spawns a ListenClientThread and a SendClientThread

ListenClientThread.java – This class is a thread spawned by Client.java, it is simply a terminal which prints out every line sent through by the server

SendClientThread.java – This class is a thread spawned by Client.java, it sends any input from the user through the socket, to the appropriate socket on the server side

On the Server side, there are four classes which control operations and two utility classes that provide definitions and manage file IO

The utility Classes are Message and User,

-User keeps track of the username and private communications of each user, incoming and outgoing

-Message keeps track of the content of each message, the user it came from, and the color the server has assigned to that user. It also contains methods for accessing each of those parts and generating an output string

The main classes are Server, ServerThread,UserThread, and UserListenThread

-Server is the main execution thread and exists primarily to create shared resources, load in any existing messages from the local directory, and spawn a ServerThread for every incoming connection (up to 20 concurrent)

-ServerThread is the thread that is spawned for each incoming connection, it creates some shared resources, spawns a UserThread and a UserListenThread, and waits for each to terminate before closing the socket and I/O streams

-UserThread is a thread which primarily consists of a loop which checks, every second, if there are new messages in the master message list ArrayList<Message> messagesList

Additionally, UserThread checks its associated user’s private communication targets, removing inactive users from the list and informing the user of each inactive user it removes

-UserListenThread is a thread which primarily consists of a loop reading each input from the user and parsing it to the correct form. It handles all ‘@command’ commands

In each server-side class, the ArrayLists containing users and messages are protected every time they are accessed or changed by two Reentrantlocks, one for each ArrayList. This ensures that no two threads can access the ArrayList<User> that holds the users or the ArrayList<Message> that holds the master message list at one time.