Final Report: Java Virtual Chat Room

Andrew Black – 250834840

Samuel Mallabone – 250844429

## **Overview**

Java Virtual Chat Room is a portable chat client that allows users to launch the program, and join a chatroom of their choice. Chat rooms can support any number of users, and users can switch between chat rooms as they please. Upon start up of the server, the Chat Room administrator is prompted to enter an integer value to set the number of available chat rooms for that specific session. Users can connect and disconnect from the server and dynamically change chatrooms as they please.

## **Program Flow**

The following bullet points will outline the life of the Java Virtual chat room, from initialization to termination.

1. Chat room admin starts up the main server, and allocates the number of chat rooms that will be available for the current session.
2. After receiving the number of chat rooms, the server is ready to accept client requests. Updating each new client window with the available number of chat rooms.
3. Users can start the Java Virtual Chat Room client, and after clicking the connect button, can choose between any of the available chat rooms.
   1. When a user selects a specific chat room, an integer is appended to the beginning of their message, which the server parses and utilizes to update chat rooms accordingly based off that value.
4. When the client wishes to end their session, they may click the disconnect button, which disconnects them from the server and then the client enters and idle state until the users wishes to reconnect or close their session.
5. Upon the administrator wishing to end the server’s session, they can type “done” into the server terminal window, which will gracefully terminate the designated user threads and terminate the server thread.

## **Needs**

For Java Virtual Chat Room, there are a theoretically infinite number of users that can connect to one, or any given chat room. In the servers case, it actually only runs on one thread. In order to create the chat rooms that are apparent to the client, the server utilizes the preceding integer values of the client’s message to post and update clients in the chat room number that match the integer value form the message. This way, the server simulates chat rooms existing on different threads, but in reality it only updates user’s message boards depending on the chat room of their choice.

## **Synchronization**

In order to keep the message board’s of each user in sync, the server thread must update each of them accordingly based off their chat room choice. As touched on in the preceding paragraph, only users currently in a chat room will receive the messages from other users posting to that message board. The server achieves this by checking the chat room that the client thread currently belongs to, and updates their message board accordingly. This way, clients will only be displayed messages from their current chat room, and upon switching rooms their message board is wiped and they begin to receive the messages pertaining to that room. In order to protect the active session information, a semaphore is used to ensure that the vector containing the client threads is only interacted with by one thread at a time. This enforces mutual exclusion between the systems threads enforcing that they don’t interfere with the critical section simultaneously which could lead to the corruption of data.

## **Termination of the Program**

The Java Chat Room utilizes a vector of the current client threads to track and manage each user’s current session. The termination process can be initiated by either the client or the server, and is handled in differing ways. If the client wishes to end their current session, they select the disconnect button or exit the client window and a shutdown command is sent to the server. This command will terminate the client’s current session, and the client’s thread within the server removes itself from the vector of current client sessions and terminates gracefully. If the administrator wishes to end all current chat room sessions, they type in a command to the server terminal window (“done”), which they begin the shutdown process. The main thread within the server call shutdown on the existing server thread, which releases the resources allocated to the server thread. Upon termination of the server thread, it’s destructor is called which goes through the current client threads, closes each of their sockets. Which allows the client threads to individually terminate their threads in a graceful manner.

## **Modifications Made**

Through out our design process, a few positive iterations were made to our overall design. The first stemmed from the poor utilization of multiple server threads operating on different ports. This was later replaced with a single server thread that was able to dynamically manage all client threads and their respective session. This was achieved by adding commands that are sent from the client sessions to the server depending on their chat room choice. These specific commands preceded user messages and allowed for the server to allocate the messages and the users to their own virtual chat rooms. The most substantial modification came from use of our own Thread class as opposed to the std libraries available one. By making this change we could clean up the user threads more effectively by utilizing the destructors of the defined server thread class.

## **Discussion**

Throughout this course the material has been engaging, and the labs were constructive and very applicable to the course content. Each lab was challenging, but not overly time consuming and in succession aided in developing the tools necessary to complete the final project. The material was interesting and we thoroughly believe that through the concepts explored within this course assisted in making us better programmers.

Note: Dan was an excellent TA who not only assisted us tremendously with our labs, but also shared incredible incite into the industry. He was always available and engaging, and was open to helping with things branching outside the course content.