ID2010VT201 – Programming of Interactive Systems

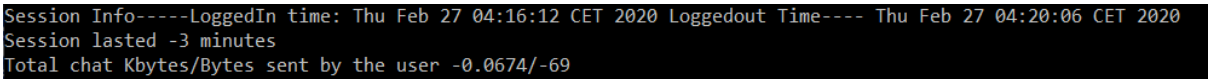
Lab 1 – Chat Programs

Carried out by:
Nagasudeep Vemula- vemula@kth.se
Akhil Yerrapragada - akhily@kth.se
27/02/2020

We have carried out the following mandatory tasks from the provided question file :-

I. Enabling clients connected to the service to see whenever clients join or leave the chat. Using the .name command a client is able change his name and this should be seen by all connected clients.

II. Chat server console prints when someone connects and disconnects. In the case of a disconnect the session time and how many messages or Kbytes that user has sent is also displayed.



III. Add an AFK (Away From Keyboard) detector that automatically notifies the other clients when someone has been inactive for a certain period of time. The message should be automatically generated when the timeout period expires. Add a command in the client for turning on and off the display of such automated messages.

Design Decisions

1. The Heartbeat signal is needed to keep track of which users are active/available on the chat client. This occurs from the client side in terms of storage and sending of the signal and no data store is present on the server side.

```
public void heartBeat(long number) {
        ScheduledExecutorService scheduler=
                          Executors.newSingleThreadScheduledExecutor();
        Executors.newSingleThreadScheduledExecutor();
        Runnable task = new Runnable(){
                public void run() {
                        try {
                                myServer.AFK(myName);
                                check.clear();
                        } catch (RemoteException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }

                }
        };

        if(!check.isEmpty())
                persist.shutdownNow();



        long delay = number;
        check.add(counter++);
        persist = scheduler;
        scheduler.schedule(task,delay, TimeUnit.SECONDS);
```

2. Timer will be defined by us and once the user is idle and the timer exceeds this time the thread execution starts and away from keyboard detection displays.
3. Data Structures present on the server end will be used to handle information regarding which user has logged into the chat client and list of active users is also retrieved from here. The server will store the list of users and return the active ones on receiving prompt.

```
protected HashMap<RemoteEventListener,Byte> userBytes = new HashMap<RemoteEventListener,Byte>();
protected HashMap<RemoteEventListener,Calendar> userLoggednfo= new HashMap<RemoteEventListener,Calendar>();
protected String text;
protected String InName;
protected List<String> usernames = new ArrayList<String>();
protected List<RemoteEventListener> disableAFKusers = new ArrayList<RemoteEventListener>();
```

4. Key value pairs are laid out in HashMap format and the messages are inserted into this as the users keep sending them. From this the number of bits stored and the size of the total number of messages sent can be obtained. For each user we return how many bits are stored from the key values.

Implementation Strategy

Interface:

```
  public void addActiveUsers( String name)
                throws java.rmi.RemoteException;


public void removeDummyUser(String name)
                throws java.rmi.RemoteException;

public List<String> sendActiveUsers()
                throws java.rmi.RemoteException;

public void swapUserName(String oldName, String newName)
                throws java.rmi.RemoteException;

public void AFKRequiredUsers(String name, final boolean status, final RemoteEventListener currentListner)
                throws java.rmi.RemoteException;

public void AFK(String name)
                throws java.rmi.RemoteException;
```

1. Heartbeat Signal- For the heartbeat signal to be implemented we have enabled it on the client end. Each time the client sends a message, a counter will be initiated on a separate thread and when the time lapses, the method will be invoked by the scheduler. The method updates all the clients that the current user is AFK.

2. Away from Keyboard- We have implemented a toggle feature for the away from keyboard messages to turn them on and off based on the user's need, this is operable using the enable and disable AFK commands. The names of the users AFK are taken from the collection of names present on the server and from this we will validate if the user called by enable or disable AFK has a match or not. For the user that has AFK disabled, we will not send the message.

3. The ". active" command accesses the collection that stores the usernames and it returns the list of active users from this list onto the client.

4. The HashMap consisting of the key value players is used to store the session information. This can be displayed on the server end and the values stored comprise the login information.

5. The session information has also been configured to consist of the following – Kb's of messages sent, logged in time, logged out time. The server is made more informative with this event log type of storage.