Chat Server and Tuple Spaces:

Problem : To implement a chat server using the synchronization mechanism of tuple spaces.

Solution:

Tuplespace: It is a mechanism for accessing concurrently the different patterns in the memory.

Actually We needed to implement a chat server and Tuples spaces for this assignment.

Implementation of Local TupleSpace:

The main purpose of tuple space here is to store the tuple (pattern) in a global storage so that all th threads which subscribes to the chat server can access the pattern (messages when talking in context of chat server)

We needed to write the get functionality and the put functionality for the Tuple server. Below is the pseudo code for the get and the put methods

get :synchronized method for finding and getting the pattern
Input: patternToSearch
output : resultingPattern

loop forever

for each string the tuple spaces
if length of pattern is equal to the string
if pattern is **null** or pattern is **equal** to the string
remove the pattern from the tuple (arraylistin our implementation)
return the matching string //resultingPattern
make the thread **wait** if the pattern is not found

put:synchronized method for writing the pattern
Input: tupleToBeWritten
output :none

write the tuple to the array list
notify all the waiting threads

Implementation of Chatserver:

We needed to implement the chat server that can acts as the interface for diffferent listners of the different channels. All the lsitners should get all the message fo that aprticular channel (at most 10 previous message) . Even the remote listeners should get all the messages.

We needed to implement **writeMessage** and **openConnection**

writeMessage: Method called by chat UI to write the messages to each listeners
Input : Channel to which it should write the message.
Ouput : None

get all the listeners for the channel
write message to each window for that channel
put the listeners back to the tuple spaces

get the min(first of the interval) and max count(last of the interval) of the message
if the **difference** of the interval **crosses** the number of **rows** the remove the first message of the interval and **increment** the interval to the next message.
Push all the **popped** tuples back to the tuple space.

OpenConnection: This method is called when ever a new listener is attached to the channel.
Input: Channel
Output: Chatlistner for that channel

give the id to this listener // in our case it is in incrementing order
get the minimum and maximum count of the window for that channel.
Put to this listener all the message from the minimum to maximum count.
Push all the pooped objects to the tuple.

In a nutshell we have stored each and every attribute into the tuple space with a pattern and we fetch these pattern whenever we need it . We are using these stored values to put the message to each listener whether at th local or the remote machine. We can imagine tuple space as the global storage from where the tuples(nothing but messages) can be fetched and chat server writes these messages to the listening windows .

Reference and Acknowledgments:
This assignment has done with the help of suggestions from the course assistants and the discussion with the colleagues.