Hannu Huhtanen
291288

This is a report on chat system that utilises a tuplespace.

---Tuplespace---

The tuplespace stores the tuples in an ArrayList and all methods that access the list are synchroniced to to pervent concurrent modifications. For storing and searching tuples there are probably better choices but the ArrayList also works. Multiple locks could have been used for different tuples to achieve better concurrent performance but I wanted to keep the implementation simple. On saving a new tuple every other thread is notified which could be avoided with multiple locks but this doesn't affect the correctness of the solution.

---Chat Server---

The chat server uses a single status tuple per channel for keeping track of message numbers and to achieve mutual exclusion. Every time a new message is written it is added to the queue of every currently registered listener. Also on new messages the channel status tuple is updated and if the message buffer is full the oldest messages are removed from the tuplespace. When a new listener is added it also locks the channel status tuple and passes all messages to the queue of newly created listener.

---Tuples used---

([Server Id], [row count and channel names])

This is the chat server status tuple. It has a predefined server id for differentiating this server's tuples from possible other tuples in the tuple space. The second spot is used for storing the row count and channel names seperated by a seperation character. This tuple is mainly used to get server settings when a client connects to a existing tuple space.

([Server Id], [Channel name], [Id of first and last message])

This is the channel status tuple. It has the same server id as the server status tuple. The next spot is the name of the channel. The third spot contains ids of the first and last message in the buffer. The channel status tuple is used often to achieve mutual exclusion of access to channel's messages. There is one channel status tuple for each channel of the server.

([Server Id], [Channel name], [Message Id], [Message text])

This is the tuple of a single message. It has the same server id as the server status tuple and channel status tuples. It also has the name of the channel it belongs to. The third spot is for the message id of the message. The id is unique compared to other messages of the same channel. The fourth spot is for the message's text. The message id should always be between first and last message ids of the corresponding channel status tuple.

---Correctness---

The implementation uses on purpose simple solutions to achieve high level of correctness. The code also passed all tests that were supplied with the starting code.

---Unexpected behaviour---

I haven't noticed any unexpected behaviour with this implementation.