Sprint 2 Team Report

Team: 201

Repo URL: http://github.ccs.neu.edu/cs5500/Team-201-SP19
Reviewer: Robert Bronstein

Score: 98

Comments

You all completed your sprint commitments well and were generally above and beyond I thought. However, there are some code quality issues I found that should not be present. The precedent set from discussion with the professor is that a grade of 100 or higher (extra credit is indeed possible) shall not be given to any team for whom there are visible/discoverable code quality issues. On the whole, I guess they are minor, but you need to bear in mind that:

1. The code itself is the product, so it needs to be polished. You are writing this both with the client in mind and the eventuality that this code will be handed off to someone else (which has in fact happened in previous iterations of MSD).

2. At a macro scale, quality issues (like missing/irrelevant Javadoc, unused variables and constructors) clutter the code substantially and add a fair amount of overhead for someone trying to understand the code, and they add a great deal of clean-up effort for later, when they could've been addressed readily at the start.

A summary of what I found is below. You all otherwise are well on your way and I am proud.

Good use of factory pattern for messages.
Empty javadoc return tag on one of the message factory methods.

ClientRunnable:

In your ClientRunnable class, it is unclear what you mean when you say that the service is used "across multiple conditions."

You mention adding up invalid user connections, but if they are invalid, why not just send an error message and terminate them on the spot rather than counting and storing them beyond the point needed to notify the user of their invalidity? If a user puts in a bad username, does this make their connection invalid? Are you going to disconnect them after a number of failed attempts to set a valid username? It also seems a little circumspect that the setName method still returns true even if the username entered was invalid as that would constitute a failed attempt to set the username.

There should be a null check for the NetworkConnection object passed to the client runnable and possibly other checks to assure the validity of the NetworkConnection object.

Enqueue Message should throw an exception for null.

I wonder if you can factor out some of the user/group null checks into a private method and simply pass in the error message to be logged.

Two methods in the ClientRunnable are marked as settable to private. If there is no valid reason that another class should be able to call those methods, then they should be.

GroupDao Interface:

Extra param tag.

UserDao Interface:
Non-javadoc comments.

Good use of singleton to keep db services unique
Good use of prepared statements.
Good thinking about inactive users, I had not thought about that.

GroupService class:
Empty Javadoc.

ConversationalMessageService:
Null initialization is redundant.
No interface.

User class:

Static constructor: I would log the stack trace or print it or something also. I don't know much about the exceptions thrown by the constructor, but I would want to see their output.
User constructor: no exception handling.
You may eventually want to build in limitations on how username and password can be set.
The group constructor only null-initializes so it can be omitted.
No exception handling in conversationalmessage constructor.

Db section:
Return tag on a constructor
You should probably at least log for the exceptions in the Db classes (even though for example, if an exception is thrown when closing the connection, is it even open anyway??).
No interfaces on db classes.

testLoginMessage is private.