Assignment 2

Due date

- CS 542: 11.59 PM EST, March ~~9th~~ 13th.
- CS 442: 11.59 PM EST, March ~~11th~~ 13th.

Submit your code as per the provided instructions. A signup sheet will be provided to you during class to setup an appointment with the TA to provide a demo of your project.

Updates

- Mon Feb 24 13:21:53 EST 2014: Assignment posted
- Fri Feb 28 17:50:55 EST 2014: Removed Driver from directory structure.

Assignment Goal

Through this assignment you will design a multi-threaded chat service.

Team Work

CS 542: You are required to work alone on this project.
CS 442: You have the option of working in teams of 2 students each.

Programming Language

You are required to program this project in Java.

Compilation Method

- You are required to use ANT for compilation of code written in Java.
- Your code should compile and run on *bingsuns* or the *debian-pods* in the Computer Science lab in the Engineering Building.

Policy on sharing of code

- EVERY line of code that you submit in this assignment should be written by you or be part of the code template provided for this assignment. Do NOT show your code to any other group. Our code-comparison software can very easily detect similarities.
- Post to the listserv if you have any questions about the requirements. Do NOT post your code to the listserv asking for help with debugging. However, it is okay to post design/concept questions on programming in Java.

Project Description

Multi-Threaded Chat

- Design a simple thread pool to handle server connections. The thread pool should have at least one method, borrowThread(...), in it. You can add more methods if you like.
- The object pool should be a Singleton.
- Design a Server that is multi-threaded. The Server should have a dedicated thread that waits on a socket. As soon as a connection is received, it should borrow a thread from an object pool to handle that connection.
- The server should obtain its port number from the command line.
- The Client should obtain the server address (hostname or IP address) and port number from the command line.
- Each message from a client should have the client's name in it. For example -- *John_Doe: How are you?*
- The server code should have a simple menu with at least the following options:
  - Send Message to all clients
  - Send message to a specific client
  - Print Messages from a client
  - Quit
- Use the above menu to print messsages on the screen from a particular chat client. You can implement the feature to send a message to a specific client, or to send a message to all clients.
- The message from the server can be sent to all the clients (so there is no need to try and send a message to a client).
- The client code should have a simple menu with at least the following options:
  - Give me a name
  - Send Message to Server
  - Print Message from Server
  - Quit
- The client can read from the socket when the user chooses the option "Print Message from Server".
- On the client side, the same thread could be running the menu and managing the socket connection with the server (reading/writing messages).
- Test your code by connecting at least two clients with the server.

Additional Requirements

- The entire communication between the server and a client, in the given session, should be saved in a string buffer at the server. A separate string buffer should be used for each client.
- Whenever the client sends a message, *BACKUP:*, the server should return the string buffer corresponding to that client. The client should use the Logger to dump it to a local file.
- During the demo, you will be asked to justify your choice of data structure(s) to save the client chat transcripts on the server side.

Some Design Considerations

- Handle all exceptions.
- Multi-threading code should be written correctly to handle race conditions, if any.
- Separate out code appropriately into methods, one for each purpose.

Code Organization

- Your directory structure should be the following:

```
-firstName_lastName
  ---chat
     ----- build.xml
     ----- README.txt
     ----- src
          ---chat
           ----------util
           -----------------ThreadPool
           -----------------All Util Files here
           ----------server
           -----------------ServerDriver
           ----------client
           -----------------ClientDriver
           ----------other packages that you need
```

Code Templates

- None provided

Submission

- Same as Assignment-1.

General Requirements

- Same as Assignment-1

Late Submissions

- The policy for late submissions is that you will lose 10% of the grade for each day that your submission is delayed.

Grading Guidelines

Grading Guidelines.

mgovinda at cs dot binghamton dot edu
Back to Design Patterns