**CS 350 Project 2: Programming with Java**

**DUE**

**10/20/2015 @end of the day**

**Total: 100 points + 10 extra credit**

# Objective

This project requires you to design and implement a Java program by using Multi-threading and Socket

# Install software before programming

JDK

Eclipse or other Java IDE of your choice

# Requirements

In this project you are to implement a Java program for a chat room. To implement this, you will need a server-client design.

The server program will do the following:

* Open a server port (the default port number is 8888), and listen to the clients’ requests. The server should be able to accept a user defined port number as the input. The server should accept at least 10 simultaneous connections.
* Once the server gets a request from the server socket (ServerSocket in Java API), it will create a client socket (Socket in Java API) from the ServerSocket by listening and accepting the connections.
* Then, an open input and an output stream for this socket will be created in client's thread since every client is served by the server by an individual thread. You can use a for-loop to handle numerous clients. Of course, you can also use a while-loop, so that we can have unlimited number of clients.

Your server program should contain a portion of code to implement the *worker* threads, which answers each individual client’s request. The *worker* thread should do the following:

* Open the input and the output streams for the current client.
* Ask the client's name (say john), and notify all the online clients that a new client (John) has joined the chat room.
* As long as the client thread enters a message, the message should be distributed to all other clients.
* When the client leaves the chat room, the *worker* thread also informs all the other online clients about his/her leaving. The leaving command is **"/quit"**.

To talk to the server program, you also need to write a client program, which connects to the chatting server. The client program should do the following:

* Open a socket on a given host and port.
* Open an input and an output stream.
* If everything has been initialized successfully, then it accept data from the user.
* If the client program get the confirmation of quit from the server, it cleans up the context by closing the output stream, closing the input stream, and closing the socket.

# Documentation and style requirements

Each **function** should have at least a line comment indicating its purpose. In addition, within each procedure, at the **beginning of each class comment**, provide **your name**, the **project number**, the **course**, and the **date**.

# Extra Credit (10 points)

Implement a graphical user interface for your client software and get 10 bonus points!

# Submission

The only way to submit your homework/project is to submit through Blackboard.

1) At the beginning of your source code, include the following text as comment:

a) Your full name and student ID,

b) Course number (CS350),

c) Due date, and

d) The following IPFW Honor Code quote and active links and your signature:

"By placing this statement in my work, I certify that I have read and understand the IPFW Honor Code. I am fully aware of the following sections of the Honor Code: Extent of the Honor Code, Responsibility of the Student and Penalty. This project or subject material has not been used in another class by me or any other student. Finally, I certify that this site is not for commercial purposes, which is a violation of the IPFW Responsible Use of Computing (RUC) Policy."

e) Your signature and date (Type your full name as a signature)

1. The source code of the java programs, both the server and client.
2. The compiled, executable jar files of both the server and client.

In addition to the Blackboard submission, you will also do a **in-class demonstration** of your program.

**NOTE: You will not receive more than 50 points if the submitted solution does not run.**

|  |  |
| --- | --- |
| Grading Rubric | |
| Proper comments | 5 |
| Meaningful variable and function names | 5 |
| Submitted all required files | 10 |
| The client program runs with no exceptions or errors | 10 |
| The server program runs with no exceptions or errors | 10 |
| The server program is able to handle multiple simultaneous connections | 10 |
| Proper communication between server and client | 10 |
| The server is able to broadcast messages to all clients | 10 |
| The /quit command works properly | 10 |
| Proper networking design. Communication between server and client must be via network sockets. | 20 |
| Extra Credit | |
| GUI | 10 |
|  | |
| Total | **110** |