ASSIGNMENT 2

DUE: Monday 12th December 2016, 2pm
FINAL GRADE CONTRIBUTION: 25%

LEARNING OUTCOMES ADDRESSED:

- Networking
- Exception Handling
- Input/Output Handling
- Multithreading Basics

MAIN TASK:
Create a chat server complete with clients that can connect to and use the server.
Using the Java programming language, you are required to create Server and Client programs.
The main scope of each program should be as follows:

- The Server program should have the following features:
  - Listen for new connections from clients.
  - Handle connections from multiple clients.
  - Respond to client requests.
  - Broadcast chat messages to all clients.

- The Client program should have the following features:
  - Send new connections to a server.
  - Accept user input and handle sending to the server.
  - Handle responses from the server.

Along with the features above, the following features must be implemented across the solution:

- A client can ask the server how long the server has been running for and receive the correct response.
- A client can ask the server how long the client has been in the chat room for and receive the correct response.
- A client can ask the server what the server's IP Address is and receive the correct response.
- A client can ask the server how many clients in total are currently connected to the chat room and receive the correct response.
- A client can ask the server for a list of request commands that can be sent and receive the correct response.
- The broadcasting of messages from each client should be handled by the server.
- After connecting to the server, the client should be asked for a username by the server.
- The client should only be able to chat in the chat room once a username has been selected.
- The username that a client selects should be unique to the chat room.
- The client should be able to send and listen for messages concurrently.
- The client should be able to handle the server going offline abruptly in a graceful way.
- The server should be able to handle the client disconnecting abruptly in a graceful way.
- The client should be able to send a disconnect request to the server and the server should then handle this gracefully.
- When a client connects or disconnects from the chat room all other clients that are connected should be sent a notification of what client has just performed that action.

- All input/output should be handled accordingly through the correct use of Exception Handling.
- Before connecting to a server a client must first be asked what the address is of the server they wish to connect to.
- When a client sends the server specific requests, these interactions should not be broadcasted to other clients in the chat room and should therefore only be visible by the client sending the request and the server.

TIPS:

- Handling multiple client connections to the server should be done through the use of a Handler Class that extends the Thread base class.
- The use of threads in the Client Class will allow you to correctly handle incoming messages and outgoing messages concurrently.
- Make use of the BufferedReader Class for incoming streams.
- Make use of the PrintWriter Class for outgoing streams.
- Store each client's name in a HashSet to make it easy to check for duplicates when a client is selecting a user name. This will also help when the server responds to a client's request for the total count of connected clients.
- Store each client's outgoing stream in a HashSet to make it easier to broadcast messages to all clients as soon as a new message is received by the server.
- If you compile your solutions into .jar files then you will be able to test running multiple client connections easier as you can have a different console window showing the output for each client and the server.
- The full solution should be in two separate Java projects as there will be two Main Methods, one for the server and one for the client.
- You should **not** implement any GUI for this project.

SUBMISSION:
You should submit the following files:

- ClientMain.java file – This should contain a main method that will instantiate a new Client Instance.
- ClientInstance.java file – This should create a new client that can connect and interact with the server in the way described above.

- Server.java file – This contains a main method and should create a new server that can accept connections from new clients. It should also contain an embedded class that extends the Thread base class to handle multiple clients concurrently. The code in this file should allow the server to interact with the client in the way described above.

- Accurately document all your files with Javadoc comments

MARKING:
This assignment contributes 25% of your final grade for this module, and will be marked according to how far the following requirements are met:

- The Java code should be laid out according to a consistent format, and it should contain clear comments
- The Java code should correctly implement the functionality set out above.
- The javadoc documentation should be full (one document comment for each class and each method of each class), clear and informative.

A first-class solution (70+%) will meet all these requirements fully; a 2.I solution (60-69%) will meet most but perhaps not all of these requirements (e.g., the code may not quite implement all the desired functionality, or may lack comments, or have an untidy layout); a 2.II solution (50-59%) will have some more serious faults (e.g., the code may fall some way short of all the desired functionality, or may contain syntactic errors); a third-class (40-49%) solution will have serious faults, though it should still show that a decent attempt was made (e.g., code that falls further short of being functional - though it still shouldn't be too far away). A solution getting a failing grade will simply be bad. Failure to hand in a solution will get a zero grade.