# COP5007 Java Programming

# Programming Project 7

### See Dropbox for due date

### Optional will replace lower or missing project grade

### Project Outcomes:

Create a Java application that:

1. Uses multiple thread without deadlocks or race conditions
2. Uses network sockets to send and receive data in a client/server format .
3. Handles exceptions .
4. Develops a program with several classes.
5. Creates UML class diagrams.

**Prep Reading**

Absolute Java textbook, Chapters 1 – 14, 17 and 19 and Java Tutorial on Concurrency and Networking <https://docs.oracle.com/javase/tutorial/>.

### Project Requirements:

1. In this project you will implement a threaded chat server and a corresponding chat client. Every message that the server receives will be sent back to every other client. Each client should start a new thread. The client allows the user to type in a message and hit enter to send it to the server.
2. The project may be implemented as you see fit, however good object oriented development (multiple cohesive classes) and good programming practices are required.
3. The project can be command line or GUI but a command line program is all that is needed.
4. Project Implementation suggestions
   1. Server
      1. Create a ServerSocket connection for the server.
      2. Create a collection(Array, ArrayList etc) that will store the clients Socket Connections.
      3. Start the server in an infinite loop and wait for client connections.
      4. For each client socket received
      5. Create a ClientHandler object (see below) that implements the Runnable interface.
      6. Create and start a new Thread on current ClientHandler object.
   2. Client Handler
      1. Using an infinite loop read any data from the socket
      2. If data received print it to the server console and resend to all clients.
   3. Client
      1. Opens a socket to the server
      2. Creates a Client Listeners (see Below) in a new thread to listen for responses from the server.
      3. Using an infinite loop sends any message to the server.
   4. Client Listener
      1. Reads any messages from the server.
      2. Prints them to the console
5. See BankService.zip for example of client/server application.
6. The program should handle all required exception gracefully. Exiting the client or the server can be handled by simply closing the connection.
7. How to run multiple clients
   1. JGrasp does not allow you to run multiple programs at the same time so you must run each client via the command line or script file.
   2. To run multiple clients from the command line,
      1. open a command line window for each client.
      2. start the client by typing: *java Client* (or whatever you called the Client class).
   3. To run from Windows Explorer
      1. Create a text file with following command:
         1. java Client (or whatever you called the Client class).
         2. pause
      2. Give the file a *.bat* extensions such as *chatClient.bat*. The .bat extension will allow you do run the file by double clicking it.
      3. Run the .bat for each Client instance you need.
8. Programmer Design information
   1. As you may have noticed much of the details for this project are left up to you.
   2. You may create any additional classes, methods (especially private utility methods) or instance fields you think may be needed, however you will be graded on good object oriented design, so points will be taken off for unnecessary classes, methods or unused instance fields or instance fields that should be local variables.
   3. There is no need to add any additional functionality such as checking for duplicate questions, or duplicate user files etc, just focus on the required functionality.
9. Create 2 UML Class Diagram for this project.
   1. Design Version - completed prior to coding the project to assist with coding.
   2. Final Version - completed after the coding that accurately represents the final version of the code.
   3. All instance variables, including type.
   4. All methods, including parameter list, return type and access specifier (+, -);
   5. No need to include the class with main method in the UML diagrams.  Refer to the UML Distilled pdf on the content page as a reference for creating class diagrams

**Submission Requirements:**

Your project must be submitted using the instructions below. Any submissions that do not follow the stated requirements will not be graded.

1. Ensure you submit all the source code, see above specification and grade sheet for details

2.  Remember to compile and run your program one last time before you submit it. If your program will not compile, the graders will not be responsible for trying to test it.

3. Follow the submission requirements posted on elearning.

**Important Notes:**

1. Projects will be graded on whether they correctly solve the problem, and whether they adhere to good programming practices.
2. Projects must be submitted by the time specified on the due date. Projects submitted after that time will get a grade of zero.
3. Please review UWFs academic conduct policy. Note that viewing another student's solution, whether in whole or in part, is considered academic dishonesty. Also note that submitting code obtained through the Internet or other sources, whether in whole or in part, is considered academic dishonesty. All programs submitted will be reviewed for evidence of academic dishonesty, and all violations will be handled accordingly.