ENSF 619 – Software Design and Architecture I

Lab Assignment #2: Client/Server Programming

Assignment Weight: 10% of the final grade

Due Dates	
Deliverable I: - Exercises 1 - 3 - Exercise 6, Task 1	**Demo:** Your design in Exercise 6-task 1 during the lab section on **Friday February 1st**. ***All group members must be present for the demo.*** Submit electronically on D2L **before 12:00 PM on Friday February 1st**
Deliverable II: - Exercises 4 – 5 - Exercise 6, Task 2	**Demo:** During the lab section on **Friday February 8th**. ***All group members must be present for the demo.*** Submit electronically on D2L **before 11:59 PM on Friday February 8th**

This is a Group Assignment. Maximum of 2 students per group.

The objectives of this lab are:

1. Concurrency (Using Thread pools)
2. Client-server communication using Java Sockets
3. Serialization and Deserialization of Java Objects
4. Experience with client-server architecture


The following rules apply to this lab and all other lab assignments in future:

1. Before submitting your lab reports, take a moment to make sure that you are handing in all the material that is required. If you forget to hand something in, that is your fault; you can't use `I forgot' as an excuse to hand in parts of the assignment late.
2. **20% marks** will be deducted from the assignments handed in up to **24 hours** after each due date. It means if your mark is X out of Y, you will only gain 0.8 times X. There will be no credit for assignments turned in later than 24 hours after the due dates; they will be returned unmarked.


Lab Marks: 75 marks + 10 bonus marks

Lab Requirements (5 marks):

The code in all lab exercises must be managed by Git. Groups must share their work via Git. This will be graded during the demos.

Exercise 1: A Very Simple Client-Server Program (8 marks)

In this exercise, you will complete the implementation of a very simple client-server program. Where client prompts the user to enter a word (in fact a string of characters), then passes the string to the server. The server is supposed to examine the string and send a message to the server indicating whether the user input is a palindrome or not. Your job in this exercise is to write only the definition of the class `Server`, the definition of class `Client` is given.

What to Do: Download the definition of the class `Client` from D2L. The client connects to the server running on port 8099 of the "localhost". The main job of the server that you need to implement is to read a message from socket, examine it for palindrome and send an appropriate message to back to the client. For your information: A string is called palindrome if it spells the same word from both ends. As an example, the word "radar" is palindrome. You can assume that user inputs are always lower case and contain only alphanumeric characters, and no punctuation. You don't need to do any error checking for these requirements.
To test your program, you should first start running the server. When server started running, it is always a good idea to show a message on the terminal or console screen that:
Server is running ...
Do not make any change to the Client class. Here are the screenshots of the outputs of the client and server programs.

Server	Client
Server is now running.	please enter a word: radar radar radar is a Palindrome. please enter a word: 121 121 121 is a Palindrome. please enter a word: java java java is not a Palindrome. please enter a word:


What to Submit: Please submit all the java files including the files you have created/modified, along with a PDF file containing a sample output of your program in a zip folder. You need to provide the Javadoc comments in your code, but you don't need to submit the HTML files.

Exercise 2: Date-time server (8 Marks)

In this exercise, you need to implement a simple client that connects to a date server. The date server class is given to you. Please download `DateServer.java` from D2L. The server runs on port 9090 of your local machine and waits for a client and when a client connects to the server, client can select to receive `DATE` or `TIME`. Depending on the client's selection, server sends the response to the client. A sample output of the program is given.

Your task is to create a client class that connects to the date server. Please do not make change to the date server class.

Server	Client
Server is now running.	Please select an option (DATE/TIME) DATE 2017-02-56 Please select an option (DATE/TIME) TIME 15:19:41 Please select an option (DATE/TIME) abcd Wrong input, please try again Please select an option (DATE/TIME)

What to Hand in: Please submit all the java files including the files you have created/modified, along with a PDF file containing a sample output of your program in a zip folder. You need to provide the Javadoc comments in your code, but you don't need to submit the HTML files.


Exercise 3: Serialization and Deserialization of Java Objects (10Marks)

This exercise focuses on the concept of Serialization and Deserialization of Java objects and will be implemented in two parts. In the first part of this exercise you are going to complete the definition of a given Java class that is supposed to read from a text file of music records, called someSongs.txt. Then it should build an object of music record and serialize it into a file called `mySongs.ser`.

In the second part of the exercise you will complete the definition of another class that is responsible for de-serializing the same object from a binary file.

Task 1 –Serialization: Download files `MusicRecord.java`, `WriteRecord.java`, and `someSongs.txt` from D2L. Read the content of the file `MusicRecord.java` carefully to understand how this class is used to represent a music record, and how its methods may help to retrieve or set the information about each music record.

Then, read the file `WriteRecord.java` to understand the content of this file. You should complete the missing parts of the given code based on the given comments in this file. Remember that you need also to make the `class MusicRecord` a serializable-class.

Once your code is complete if you run your program it should read the text file called someSongs.txt, and create a binary file called `mySongs.ser`. The created output file is expected to contain objects of music records, which is a binary file, and if you try to open it with a text editor, you will find it almost unreadable.

The only way to browse the content of this file and to find out what it really contains, is to write another program that reads the content of `mySong.ser`, and de-serializes the records.

Task 2 - Deserialization: Download the file ReadRecord.java from D2L, and read it carefully. Then, complete the missing parts that are supposed to de-serialize the objects from `mySongs.ser` and displays them on the screen.

Once you completed your code and your read, test your program using the object file `mySongs.ser`. It should display the same records that exists in someSongs.txt.

If your first test with `mySongs.ser` was successful, then download the file called `allSongs.ser` (from D2L (if haven't already done) and run your program with this file that contain a different number of records.



What to Submit: Please submit all the java files including the files you have created/modified, along with a PDF file containing a sample output of your program in a zip folder. You need to provide the Javadoc comments in your code, but you don't need to submit the HTML files.

Exercise 4: Tic-Tac-Toe with a Thread pool (22 Marks)

Note: You can combine Exercise 4 and 5 together. But it may be easier to complete exercise 4 and then attempt to build the GUI in exercise 5.

In this exercise, you should use the first version of your Tic-Tac-Toe Game and you should modify it in a way that it can support multiple games simultaneously. Each game will have its own thread and the communication between players is managed via sockets.

One obvious step that you need to do in this exercise is to define a class that represents the client-side of the game (a client user interface). This class is not only responsible to take the player's inputs and pass them to the server-side of the game, but also is responsible to receive the opponent's inputs from server-side and display them on the board.

The client side must run in two different terminal-windows and a game should not start before two players are available (must be monitored by the server). The server-side must use a threadpool to start a new game for every two clients that connect.

For the server-side one possible option is to create a new class for the server and make your existing class `Game` implement `Runnable`. This means the server class should include data fields such as `ServerSocket and ExecutorService`. It must create and run a new game when two clients (players) ask for a connection and those connections are accepted.

What to Submit:
Please submit all the java files including the files you have created/modified, along with a PDF file containing a sample output of your program in a zip folder. You need to provide the Javadoc comments in your code, but you don't need to submit the HTML files.

The following page shows a few screenshots of the program.

Screen Shot from Server Treminal

```
Server is running...
```

Screen Shots from X-Player Terminal

```
          |col 0|col 1|col 2
          +-----+-----+-----+
          |     |     |     |
    row 0 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 1 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 2 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
Message: WELCOME To THE GAME.
Get the name of the X player:
```

Screen Shot from O-Player Terminal

```
          |col 0|col 1|col 2
          +-----+-----+-----+
          |     |     |     |
    row 0 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 1 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 2 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
```

```
          |col 0|col 1|col 2
          +-----+-----+-----+
          |     |     |     |
    row 0 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 1 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 2 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
Message: WELCOME To THE GAME.
Get the name of the X player:
Mike
Message: Waiting for opponent to connect
```

```
          |col 0|col 1|col 2
          +-----+-----+-----+
          |     |     |     |
    row 0 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 1 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 2 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
Message: WELCOME To THE GAME.
Get the name of the O player:
Judy
```

```
bin — java — 80×24
Mike what row should your next mark be placed in?
1
Mike what column should your next mark be placed in?
1
          |col 0|col 1|col 2
          +-----+-----+-----+
          |     |     |     |
    row 0 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 1 |     |  X  |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 2 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
```

```
bin — java — 80×24
Judy it is your turn to make a move.
          |col 0|col 1|col 2
          +-----+-----+-----+
          |     |     |     |
    row 0 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 1 |     |  X  |     |
          |     |     |     |
          +-----+-----+-----+
          |     |     |     |
    row 2 |     |     |     |
          |     |     |     |
          +-----+-----+-----+
Judy what row should your next mark be placed in?
```


Exercise 5: Tic-Tac-Toe with GUI (12 Marks + 5 bonus possible bonus marks) Multi-player client server version with a GUI

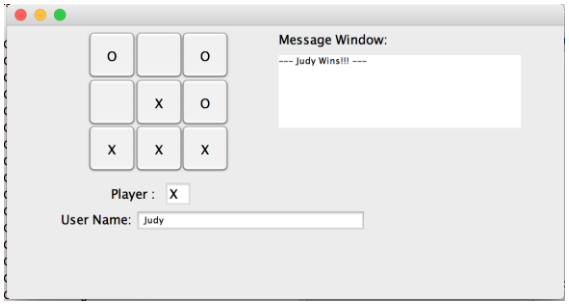
Create GUI (similar to your assignment in ENSF 519) for the clients that allows players to use their mouse to mark the board with an X or O. Each game is to be played by 2 clients; i.e. OPlayer, and XPlayer. Each client must have their own GUI. This is because each client runs on a machine.

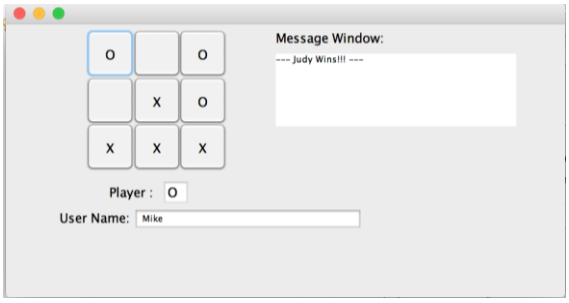
Here are the minimal functionalities of the component of your client window/frame must have:

- A presentation of the board
- A text area that displays the game messages for the communication or information exchange.
- A text box that players can enter their name.

Note: You have complete freedom to design your GUI for this game differently. You may also choose to have additional components/events in your GUI.

Here is a screenshot of the GUI I had for x-player and o-player clients:





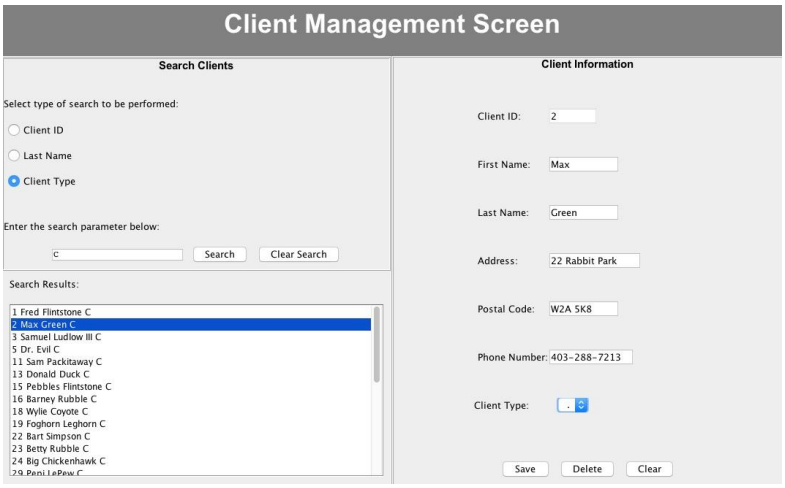
Bonus marks: Having the client(s) and the server running on different machines. At least two machines; one server and one for the clients.

What to Submit: Please submit all the java files including the files you have created/modified, along with a PDF file containing a sample output (i.e. screen shots of your GUI) of your program in a zip folder. You need to provide the Javadoc comments in your code, but you don't need to submit the HTML files.



Exercise 6: Modifying Software Architecture (22 Marks + 5 possible bonus marks)

In ENSF 519, Project I, exercises 2 – 4, you built a **desktop application** for a Client Management System:



In this exercise, you are asked to modify the architecture of this software to a client-server architecture. The requirements of the system are as follows:

- Your server must be able to support multiple clients (i.e. implement a thread pool)
- You must use the MVC design pattern
- You must use object serialization for communication between the server and clients
- Your design must closely follow the SOLID Principles

Task 1 – Design: Create a high-level class diagram. Your design must clearly show the different packages for client and server side of the system. The MVC design must also be clear in your diagram.

Task 2 – Development: Implement your design from task 1.

Bonus marks: Having the client(s) and the server running on different machines. At least two machines; one server and one for the clients.

What to Submit: Submit a .pdf file for task 1. For task 2, submit all the java files including the files you have created/modified, along with a PDF file containing a sample output (i.e.


screen shots of your GUI) of your program in a zip folder. You need to provide the Javadoc comments in your code, but you don't need to submit the HTML files.

How to submit: Include all your files for the post-lab section in one folder, zip your folder and upload it in D2L before the deadline.