Assignment 2 2024/25: Java GUI-based Client/Server Sensor Aggregation Draft

In this assignment you are going to develop a Java client/server application that sends and receives data to/from a PC and an embedded Linux device such as the BeagleBone Black (BBB) or Raspberry Pi (RPi). This assignment can be completed entirely on a single desktop machine and all tasks are implemented in software alone.

Please send **ALL** questions on problems that you are having to the discussion forum on the module Loop page. The assignment is worth 15% of your overall mark in the module.

To uphold DCU's academic integrity policy, assignments **must** adhere to the following conditions for validation:

1. **Utilise the provided client/server code** as the foundation of your submission; failure to do so results in a zero mark. While modifications are allowed, the overall structure must be maintained.
2. **Create your submission using the course notes**. Ensure that coding structures and styles align with the examples in the notes. Significant deviations may lead to an interview and/or a zero mark.
3. **Include an accompanying video demonstration** with your submission. Assignments lacking this video will be marked as zero.

Assignment Task: Java GUI-based Client/Server Sensor Aggregation

In this assignment you must build a client/server application, where multiple GUI client devices (simulated embedded devices) send simulated environmental sensor information to a central server where it is aggregated and displayed using a GUI.

Base your assignment on **three** types of environmental sensors that you choose from the following list only: light level, pressure, CO2 level, humidity, particle pollution level, temperature, sound level, wind speed, and methane level.

The Simulated Sensor Device Client Application

In this application each client application should run on the desktop/laptop device. You will need to run the same sensor device application multiple times as different instances on the same machine to simulate multiple sensing devices. You can do this from within Eclipse, or you can use different command prompt/terminal windows if it makes it easier for you to manage.

Figure 1. Based on this description, Leonardo.ai generated the prototype above.

The sensing device client should have the following features:

- You should pass the server address (e.g., localhost or 127.0.0.1 to connect to the same machine) and the sensing device name to the client application as command line arguments when you first execute it. The port number can be hard coded at a value of your choosing.
- The application should have a pop-up dialogue window that allows you to change the name of the sensing device as the application is running.
- The client application should have a basic Swing GUI that displays the sensor device name and allows you a mechanism of pushing buttons and/or moving sliders to generate sensor readings that will be sent to the

server. The nature of this interface is deliberately vague to encourage variation in the solutions developed.
- A single sensing device application should have three different simulated sensor type sensors. See the required sensor types above.
- The client application should send an update every 5 seconds, regardless of whether the sensor value is changed.
- Add configurable random noise to your sensor values to improve the simulation reality.
- The X button on the window frame should disconnect the server and close the sensor device application gracefully.
- Add two novel features of your own design.

The Graphical Display Server

In this assignment the desktop/laptop PC is the graphical display server. The Java GUI application should be run by the user and remain running until it is shut down gracefully. The GUI should display sensor information using graphical dials and charts that it receives from multiple Java client devices.

The server application should have the following features:

- On execution, it should open a server socket and wait for connections (on port of your choosing) from multiple sensor device clients.
- The server should have a GUI that is built using Java Swing. It should display the data using a combination of standard Swing components and graphical components that you create yourself (e.g., based on the Canvas examples in the notes). See the next set of bullet points.
- The server should be able to accept connections from many sensor devices simultaneously (limit this number to 3 for the purpose of testing).
- The server should be able to perform basic statistical analysis on the data received, including averaging instantaneous readings from multiple sensors and providing time averaging of historical readings (e.g., moving average). Limit analysis to a maximum of 10 historical values.
- Base your assignment on the **three** types of environmental sensors listed above that you have chosen.

Please treat ‘time’ as simple N second windows. There is no requirement to feature ‘wall clock time’ in your application as it will over complicate the task. N can be any integer number of your choice and can be hard coded into your client and server applications.

The Server GUI should have the following features.

- The names of the currently connected sensing devices should be displayed along with their connection status.
- The GUI must display the last 10 average sensor values (averaged across sensor devices) over your time windows. Marks are awarded for more sophisticated displays but simple displays using standard Swing components are acceptable.
- The GUI must include a user-designed gauge graphical component that combines instantaneous readings from multiple sensors of the same type. You can replicate this gauge for each of the three sensor types. The gauge should display the average, lowest and highest instantaneous reading.
- The X button on the Window Frame should shut down the server gracefully.
- Add a novel feature of your own design.

Remember that the server can connect to multiple clients at the same time so design your client/server applications appropriately. For example,

- Structure your code appropriately to allow the server to hold multiple connections open.
- The client/server pair should **send and receive sensor objects of a class of your own design**. Proper design of sensor message class will greatly simplify the complexity of this assignment. While a basic approach might appear to be parsing string data, it will make your life a misery. I strongly suggest you review the notes on object serialization.

Provided Code

I have provided you with template code and it **MUST BE USED AS THE BASIS OF YOUR SOLUTION**. Do *not* use RMI or any other Java Messaging Protocols, only use the template code provided for network communication. This code is in the following files:

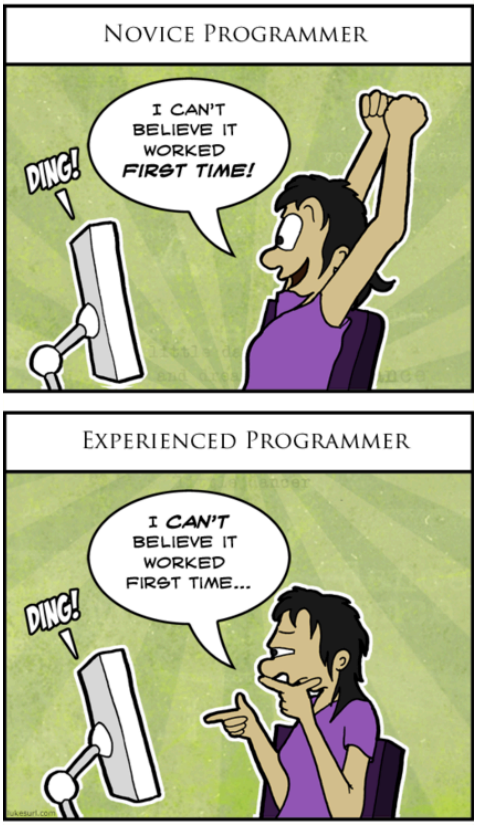
- ThreadedServer.java
- ThreadedConnectionHandler.java
- DateTimeService.java
- Client.java

This code is fully threaded and should work in its current form. You must adapt the code to the application described above, including the inclusion of a GUI. The Web browser example in the notes describes how you can attach a GUI to a client application and it is a good starting point for your assignment. I recommend that you only begin the server GUI after you have practised with developing a GUI for the client.

Testing your Solution.

You should demonstrate that your architecture is working correctly by simultaneously connecting at least two clients to the server simultaneously. Show screen captures of your system in action in your report and also demonstrate multiple-client connections in the short video demonstration that you submit along with your report.

Marking



You will receive marks for implementing the features above. You will gain marks for making the server properly threaded, for sending suitable messaging objects and for a working solution. You will gain marks for novel/extra features as outlined in the specification.

The assignment is worth 15% of your final result. The marks will be allocated against the features outlined above. To explain how you achieved these points, you should describe how you achieved these points using the section headings: Design, Implementation/Coding and Testing:

- **Design** - The overall design and features of your system. This section should be brief and should include your class structure, client/server communications mechanism etc.
- **Implementation/Coding** - The implementation of your design and the quality of your code for both the interface/client and server. **Most of the marks are allocated for this component (see below).** This section should include screen grabs of your program in action and discuss any extra features.
- **Documentation/Testing** - Short section that links to your source code, lists file names and a discussion on how to run your program. Discusses any tests you performed.

The final report should describe your design, features, interesting code segments, your messaging format etc. It should include screen grabs of your client/server in action (use ALT-PrtScr to grab a window view).

An example breakdown of marking is as follows [100%]. Please note that this is only approximate:

- **Server 25%**: Graphics display, GUI, business logic, novel Features?
- **Client 25%**: GUI, Threads and novel Features?
- **Messaging Class 15%**: Use of user-defined class for messaging?
- **System Test 5%**: Does it work correctly with multiple clients?
- **Overall Design 15%**: Good of OOP, avoids cut-and-paste code, correct use of classes, code in in correct places?
- **Documentation and Video 15%**: Has architecture description, well described, screen grabs, good description of what works, what doesn't work, conclusions etc.

The final marking scheme will differ from and there will be crossover. For example, the video demonstration will contribute marks to documentation and provide validation of the extent of your client/server implementation.

Submission Instructions

Please review the warning at the beginning of this assignment description and ensure that you include the three following elements in your submission. Your submission should include a PDF report, a screencast recording video, and a source code file (backup).

Report

Submit an electronic report in PDF format. Use the following headings in your report:

1. **Introduction**. Briefly introduce the problem.
2. **Design**. Describe your overall design and any individual plans to meet the assignment brief.
3. **Implementation**. This should be the most substantial discussion and should describe what you have achieved, how you achieved these outcomes and should discuss novel features, your user-defined messaging classes etc. It should be broken down under the headings: Server, Client, and Communication.
4. **Testing**. Describe any tests that you performed. Does it work correctly with multiple simultaneous clients? What are the limitations? Under what conditions does it fail?
5. **Conclusions**. What did you learn? What was good and bad about this assignment? What would you change? Etc.
6. **Appendices**. Include the source code as an appendix (in the same form as Assignment 1 -- i.e., label file names).

Video Demonstration:

You must include a 2-5 minute MPEG4 (h264/h265) screencast demonstration of your application running on your computer. Narrate the video to describe the client server functionality and to demonstrate communication.

Record the desktop using free screencast software -- do not use your mobile phone camera to record the video. Examples include:

- Zoom allows you to host a meeting of one and self-record.
- OBS Studio (Windows, Mac, Linux): https://obsproject.com/
- Flashback Express (Windows): https://www.flashbackrecorder.com/express/
- ScreenRec https://screenrec.com/screen-recorder/
- ShareX (Windows): https://getsharex.com/

If you have any different recommendations or experience issues with any of the above, please post to the forum. Please note that Loop has a maximum file upload of 50MB (for context, my weekly videos run at about 200 MB per hour).

The video should emphasise the client GUI, **server GUI** and show definitively the **network communication** taking place between the clients and the server. After that you can discuss threads, novel features and other required elements of the assignment, as described above.

Source Code

The source code should also be submitted so that the tutors can review it if there is a query about the code in your report. This is a secondary copy and the code in the appendices of the report is the primary copy.

- The code should be functional (e.g., comment out features that are crashing your application). Please provide instructions for the tutors on how to run your assignment. Please include the .java and .class files.
- Place the code in one 7zip (https://www.7-zip.org/) or rar format file and uploaded to the EEN1022 Assignment 2 Submission in Loop. (You can safely ignore any file format warnings in relation to the Loop submission.)

Deadlines

The assignment is due for:

The assignment is due for Friday the 29th of November, 2024

(i.e., the end of Week 12)

Late submissions will be reduced in the following way:

- 5% (gross) reduction for 10 minutes to 1 hour late.
- 10% (gross) reduction for 1 hour to 24 hours late.
- 10% (gross) additional deduction for each day late[1].
- No submission permitted 7 days after the assignment deadline.

Please note that you can submit your work multiple times before the deadline, which will help you avoid late penalties.

[1] Negative marks limited to 0% -- e.g., a score of 40/100 will receive a mark of 0% after incurring a penalty for being 5 days late.

I will discuss the assignment in more detail over the next few weeks. But please start working through the Java notes, particularly in relation to the user interfaces sections.

Yes, it likely sounds difficult but you will get through it. I will be giving out additional tips on how to complete this assignment during the coming weeks.

Plagiarism

You may work with others to understand the course notes and assignment instructions, but as soon as you work together (including with a third party) on the assignment itself then you are crossing the line.

You may use ChatGPT or other LLMs to understand the materials, but they should not be used to generate code or report documentation that you submit for your assignment.

You may be interviewed on your submission with or without cause. The interview will require that you explain how you arrived at your solution and the decisions you took (not what your code does), so please retain any draft development work and/or early versions in case they are needed. The two assignments include formative work that will also appear in the final examination, and an interview will be likely if the quality of the CA work is significantly stronger than the knowledge demonstrated in the examination for the same topics.

Please note that by the electronic submission of your assignment work I am assuming that you are familiar with the academic rules for assignments.

The assignment should be 100% your own work. If any unreferenced work is used from any other source (including colleagues, the internet, hired 3rd parties, or past students) it will be a serious matter and referable to the disciplinary board. See:

http://www.dcu.ie/info/regulations/plagiarism.shtml

All assignments for this module will be uploaded to plagiarism detection software where the content will be analysed and compared to other assignments from this year and previous years.

Grading

If after you receive your mark you are concerned that there is an error with the correction, **please contact the module tutor that corrected your assignment in the first instance**. If the tutor checks it and you are still

unhappy with their update/explanation then contact me directly and I will review the decision.

If I have failed to address any aspect of this assignment then please post your questions to the forum and not to me directly as others likely have similar questions. As always, contact me directly if your query is of a personal nature.

Good luck!