420-511-VA WEB SERVICES – FALL 2024
VANIER COLLEGE
TEACHER: SLEIMAN RABAH

LAB #1

DUE DATE: as scheduled on LÉA

TOPIC

TCP Socket Programming in Java: Implementing a Data Exchange Protocol

INSTRUCTIONS

- This is an **individual, closed-book** Lab.
- You are not allowed to collaborate and use the Internet unless explicitly instructed.

OBJECTIVES

- To understand the fundamentals of IPC message passing mechanism.
- To design and implement request and response messages.
- To implement a client-server application using the TCP Socket API.
- To understand server-side and client-side error handling.

RESOURCES

1) ServerSocket class
2) Client Socket class

OVERVIEW

In this lab, you will learn the basics of TCP socket programming in Java and apply the fundamentals of inter-process communication and message passing by implementing a pair of ***client*** and ***server*** programs.

You will implement a simple, ***text-based protocol*** that allows your client to exchange messages with your server. A text-based protocol (that is, plain text protocol) is a communication protocol whose message representation is in human-readable format.

You are provided with a working client/server implementation sample on which your code must be based.

This lab consists of ***two parts***.

PART I: IMPLEMENTING A SIMPLE TCP-BASED PROTOCOL

In this part, you will design a set of messages to be exchanged between the client and server. You must define the commands to be used to send data as well as the format (that is, the structure) of the messages to be exchanged.
The commands are the requests that your client can initiate, and your server can handle/process.
For each request initiated by the client, there should be a response produced by the server.

→ **Example** of requests that can be defined in your protocol: compute a given task, get current date and time, get the server's favorite drink, number of requests handled, etc.

REQUIREMENTS

1) Open and run the provided projects in Apache NetBeans. You must run the server first then the client.
2) Examine closely the implementation of the provided protocol.
3) Design at least ***three different*** request messages of your choice. The request messages must include the following sections: 1) a header, used for including command(s); and 2) a body. The body section must have more than one line.
4) For each request message you designed in Step 3), design a ***response message*** containing **two sections**: **1)** a header which holds the status of the message (an error code and error message). The error message must span across multiple lines;
   and **2)** a body for holding the payload of the response.
5) Change the port number that the server listens to. Use a number that is between 49152 and 65535.
6) Implement both the request and response logic of the new request/response messages you designed.
7) Write test methods for testing the implemented messages.
8) Document your implementation as well as the messages you designed.

Note: When designing the messages, you should consider two aspects: the type of messages, and the structure of messages (**see Appendix Section** below for more details).

PART II: SERVER DEPLOYMENT

Before you do this part, you **must first submit** your code through LÉA.

In this part, you are required to deploy your server on another lab computer. You can share your server implementation with one of your classmates. Once deployed, make the necessary changes to the client's implementation. Run your client and test the interactions between the client and server.

The interactions must include requests for triggering server-side errors.

EVALUATION CRITERIA

		POINTS
Functionality	Compliance of the implementation with the stated requirements.	70
Correctness	No errors, the program works well and meets all the specifications. Ability to perform as required and produce correct output.	10
Code readability and good programming practices	Programming style, good naming convention, and clarity. The use of meaningful self-explanatory names for identifiers (classes, data members, variables, constants, methods, etc.).	10
Documentation	Description of purpose of program. Use of Javadoc comment style for all methods and fields, comments on non-trivial steps in all methods.	10
	Total	**100**

WHAT TO SUBMIT

➢ Compress both the client and server NetBeans projects and upload the .zip file to LÉA.

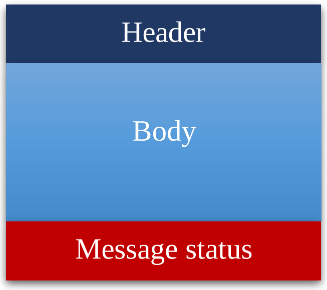
APPENDIX

Instructions about the sections of the message to be designed.

Header Section:

Contain structured fields describing the operation to be performed and optional meta data:

- Message type
- Command
- Body size
- Recipient information
- Sequence information
- Retransmission count
- Etc.



Body:

The actual data to be transmitted. That is, the data payload.

Message stauts:

Contains fields describing succesful operation or encountred errors.

- Status code and Error message