Homework 10: External Communication

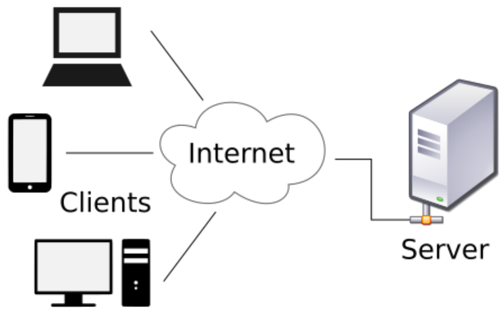
Release date: 3/25
Due date: 4/01 by 11:59 PM

Goals

- Learn how to apply the client-server method
- Learn how to use I/O streams
- Learn how to use sockets

Introduction

Whether you realize it or not, servers, with the help of the internet, power our world. The applications you used to access this handout, or post photos of your friends and family are all powered by servers, and more specifically the client-server model.



In this model, a client sends a request to the server, and attempts to carry it out. The "language" that the request must be in is often referred to as the protocol of the server. If the request does not follow the protocol, it won't be understood by the server.

Below you will implement a server that performs simple arithmetic, and can handle multiple clients at the same time. You will have to parse each client's request, and ensure that they follow the proper protocol.

Description

The following files have been provided as skeleton code:

➔ **`ArithmeticProtocol.java`**
➔ **`ArithmeticRequestHandler.java`**
➔ **`ArithmeticServer.java`**

➔ **`ArithmeticClient.java`**

A valid request to the server is a **`String`** of the form **`OPERATION NUM1 NUM2`**. **`ArithmeticProtocol.java`** contains valid operation names, as well as a constant representing an illegal request. If the request is illegal, you must send an error message back to the client. These error messages have been declared for you as constants in the file **`ArithmeticRequestHandler.java`**, which is where you will be completing this assignment.

In the file, you must implement the **`run()`** method, and interact with the client through the **`clientSocket`** field. As long as the client keeps sending requests, you must keep the connection open, and serve them to the best of your ability. You may use any I/O class that you are comfortable with in order to communicate with the client, such as **`Scanner`** and **`PrintWriter`**. Responses sent to the client must also be of type **`String`**. Example interactions can be seen below.

`ArithmeticServer.java` and **`ArithmeticClient.java`** have also been provided, and can help you test your implementation. Be sure to run **`ArithmeticServer.java`** first, as the client must have a server to connect to.

Example Interactions

Request sent to the server: **`yo`**
Response sent to the client: **`ILLEGAL_REQUEST: requests must include an operation name, and two operands all separated by spaces`**

Request sent to the server: **`yo 5 5`**
Response sent to the client: **`ILLEGAL_REQUEST: the operation name must be part of the protocol`**

Request sent to the server: **`ADD Java 5`**
Response sent to the client: **`ILLEGAL_REQUEST: the first operand must be a valid integer`**

Request sent to the server: **`SUBTRACT 53 var`**
Response sent to the client: **`ILLEGAL_REQUEST: the second operand must be a valid integer`**

Request sent to the server: **`MULTIPLY 10 10`**
Response sent to the client: **`100`**

Request sent to the server: **`DIVIDE 0 25`**
Response sent to the client: **`0`**

Submission

You are required to complete one class, **`ArithmeticRequestHandler`**, that follows the specifications outlined above. It is to be held in the file included in the skeleton code.

Submit all of the files to Vocareum ***through Blackboard***. Keep in mind that only your latest submission will be considered.

Grading Rubric

- **`ArithmeticRequestHandler`** class — 100 points total
    - 5 points each
        - Handles illegal requests
        - Handles addition requests
        - Handles subtraction requests
        - Handles multiplication requests
        - Handles division requests