Lab 5

Purpose:

- To gain experience with a non-Python network library
- To work with Java `Stream` classes
- To work with Java bit operations and the byte type

Handin:

Hand in the following files to the appropriate Dropbox folder in D2L:

- `ServerA.java`
- `ServerB.java`
- `ClientA.java`
- `ClientB.java`

Specifications:

You are to write two new versions of Lab 2, but written in Java using TCP instead of UDP.

To accommodate the change to TCP, we are adding a step to the protocol; the servers will start with a "`READY`" message to the client. The rest of the protocol is the same, however: the client sends the data bytes to the server, the server does the calculation and sends the 4-byte response, and then they both close the connection. (The client then quits, while the server loops back to accept a new client.)

To invoke the Version A client, then, would look like this:

```
ics200.lab5.ClientA localhost 10101 + 1 2 3 4 5
```

(Of course, this assumes the server was invoked using the port 10101. The server remains having only a single command-line parameter, namely the `port`.)

The difference between the two versions will be the classes used to send and receive data.

Version A

This version will most closely correspond to Lab 2 where all data was sent and received as byte arrays. You will use a single `BufferedOutputStream` and a single

`BufferedInputStream` to send and receive all data. Don't forget to `flush()` your output buffers when you're ready to actually send data over the network!

Take special care when working with larger numbers as well as negative numbers! Java assumes all bytes are signed integral values, and this can cause problems when "promoting" to integers when using "`|`", "`>>`" and "`<<`"!

The classes you create for this version are:

- `ics200.lab5.ServerA`
- `ics200.lab5.ClientA`

Make sure you name your file correctly, paying particular attention to the capitalization! All Java conventions must be followed.

Version B

This version makes better use of the library classes offered by Java. In particular, you are to use a `PrintWriter` and a `BufferedReader` to send and receive text, a `BufferedOutputStream` and a `BufferedInputStream` to send and receive the request data byte array (same as Version A for this part), and a `DataOutputStream` and a `DataInputStream` to send and receive the final calculated integer result.

Note that although you are creating a number of `Streams`, `Readers` and `Writers`, they all are just wrappers around a single `InputStream` and a single `OutputStream` that are both attached to the `Socket`; they do the work of managing the underlying byte array, so you can work at a higher level of application data. The value of this should be especially apparent when sending and receiving the integer result!

The classes you create for this version are:

- `ics200.lab5.ServerB`
- `ics200.lab5.ClientB`

Make sure you name your files correctly, paying particular attention to the capitalization! All Java conventions must be followed.