SE 424: Distributed Systems	**Recitation 1**
Semester 1 5785	**3 November 2024**
Lecturer: Michael J. May	**Kinneret College**

Network Programming Introduction

In this recitation we will begin to learn the fundamentals of network programming in Java.

1 A Simple Server and Client

We will begin by developing a simple client and server which communicate over TCP. The work the server does is trivial, but it represents a first step towards a framework for simple network communication.

The client's behavior is as follows:

1. It receives the network IP address and port of the server.
2. It asks the user for a sentence to process.
3. It sends the sentence to the server over a TCP connection.
4. The server sends back the sentence's length in characters and returns it in upper case.
5. The user chooses to continue or finish.

The server's behavior is as follows:

1. It receives a network IP address and port to listen on.
2. It listens for connections.
3. When a connection arrives, it listens for a sentence to process.
4. When a sentence arrives, it calculates its length and converts it to upper case.
5. It sends back the sentence in upper case.
6. It sends back t the length of the sentence.
7. It goes back to listening for a new connection.

We'll begin with a simple server which only can handle one client at a time. Then we'll expand the functionality to enable multiple concurrent sessions using threads.

2 Useful Classes

You'll find the following classes useful for implementing the client and server in Java:

1. `Socket`: Represents a connection socket at the client and at the server.
2. `ServerSocket`: Represents the welcome socket at the server
3. `PrintWriter`: Wraps an `OutputStream` and allows for easy writing to it using text. Don't forget to use `flush()` to force data to be sent on it if you write only a short amount of output.
4. `BufferedReader`: Wraps an `InputStreamReader` that in turn wraps an `InputStream`. Useful for easy reading of text from a socket interface or the keyboard (`System.in`)
5. `Thread`: Class that you extend to create a new worker thread. Use the `start()` method on it to get it running (**do not** call `run()` directly since that's almost never what you want).

3 What to do

After we write the code for the basic server and client in Java, your job will be to expand it to allow the client to submit more than one sentence per session. When the user enters a blank line, stop the session and close down the connection.

4 Adding Unit Tests

Use JUnit 4 to add unit tests to your server and client classes. The unit tests must exercise every method in the classes, testing how they work under correct and incorrect circumstances.

Design the methods of the classes so that they are amenable to testing using JUnit. That implies writing methods that take parameters and return values, not just using STDIN and STDOUT for input and output processing.

