Assignment 2 (2nd half of semester)

Object-Oriented Programming III (CT3535), Year 2017/2018, Semester 1

- ***Submission deadline (strict): Wednesday, 15th November, 23:59.***
-
-
-
- Use Java comments to explain your code. Missing or insufficient comments may lead to mark deductions.

Question [max. 100 marks]

You are given the following application-layer networking protocol for a simple client/server application. A client can send tokens (words) to the server which adds them to a global tokens list in memory which has a maximum capacity of different tokens. There shall never be duplicates in the list. Also, the tokens list should be sorted in lexicographical order at any time. There are three types of requests (SUBMIT, REMOVE and QUIT) from client to server.

The protocol in detail:

- Client request (a message send from client to server): **`SUBMIT`** ***`token`***

  Server: If the tokens list is not yet full, add the token to the global list of tokens (if the list doesn't contain that token yet) or ignore it (if the submitted token is already in the list). In both cases, respond by sending message **`OK`** to the client. If the list is already full or some other error occurred, respond to the client with message **`ERROR`** ***`m`*** where is some meaningful description of the error.

  `token` stands for any string which doesn't contain whitespace.

  The global tokens list should be sorted in lexicographical order . Choose an appropriate data structure from the Java API for representing the tokens list (hint: `TreeSet`).

- Client request: **`REMOVE`** ***`token`***
  Server: If the global tokens list contains ***`token`***, remove it from the tokens list and reply to the client with message **`OK`**. Otherwise, reply with **`ERROR`** ***`m`*** where is some meaningful description of the error.

- Client request: **`QUIT`**
  Server: Ends the connection to that client. No response.

An example network interaction: : SUBMIT aaa → : OK → : SUBMIT ccc → : OK → : REMOVE aaa → : OK → : REMOVE aaa → : ERROR … → Client: SUBMIT aaa → : OK → : SUBMIT ccc → : OK → : SUBMIT bbb → : OK → : QUIT

<u>PTO</u>

Create a Java program for a socket-based server which can interact with clients via the Internet[1] in compliance with the given protocol.

Your server is not required to handle multiple clients concurrently (i.e., at the same time). However, it should accept multiple clients one after another (once a client has disconnected from the server, the server should accept a new client, and so on until the server program is stopped).

Remember that there should be only one global list of tokens which is not cleared when a new client connects.

Do not create a client program. Familiarize yourself with functionality and code of the given client program ***TokenClient.java*** on Blackboard (under "Assessment") and use it to test your server.

[1] Server and client need to interact via sockets and TCP/IP, but for the purpose of this assignment they should actually run both on the machine (see "localhost" in lecture), that is, the IP-address used to contact your server should always be 127.0.0.1