Java back-end developer test

Purpose

The goal of this test is to provide us with a full understanding of your coding style and skills. We'll pay particular attention to:

The code structure
Consideration of concurrency issues
The design
Choice of data structures
Quality and use of unit tests

The goal is not to get a solution covering all special cases in a 100% robust way; the functions should be error free when used correctly but our main goal is to understand your approach to the problem.

Description

Write a HTTP-based mini bids back-end in Java which registers bid amounts for different users and levels, with the capability to return top bids per item. There shall also be a simple login system in place (without any authentication...).

Set up a Git repository with the source code, with instructions on how to execute it. Pass us the link by email

Nonfunctional requirements

This server will be handling a lot of simultaneous requests, so make good use of the available memory and CPU, while not compromising readability or integrity of the data.
Do not use any external frameworks, except for testing. For HTTP, use com.sun.net.httpserver.HttpServer.

There is no need for persistence to disk, the application shall be able to run for any foreseeable future without crashing anyway.

Functional requirements

The functions are described in detail below and the notation means a call parameter value or a return value. All calls shall result in the HTTP status code 200, unless when something goes wrong,

where anything but 200 must be returned. Numbers parameters and return values are sent in decimal ASCII representation as expected (ie no binary format).

Users and levels are created “ad-hoc”, the first time they are referenced.

Login

This function returns a session key in the form of a string (without spaces or “strange” characters) which shall be valid for use with the other functions for 10 minutes. The session keys should be “reasonably unique”.

```
Request: GET /<userID>/login
Response: <sessionkey>
<userid>: 31 bit unsigned integer number
<sessionKey>: A string representing session (valid for 10 minutes).
Example: http://localhost:8081/4711/login --> UICSNDK
```

Post an user's bid to an item

This method can be called several times per user and item and does not return anything. Only requests with valid session keys shall be processed.

```
Request: POST /<itemID>/bid?sessionKey=<sessionKey>
Request body: <bid>
Response: (nothing)
<itemID>: 31 bit unsigned integer
<sessionKey>: A session key string retrieved from the login function.
<bid>: double
Example: POST http://localhost:8081/2/bid?sessionkey=UICSNDK (with the
post body: 3.1)
```

Get top bids for an item

Retrieves the top bids for a specific item. The result is a comma separated list in descending bid order. Because of memory reasons no more than 15 bids are to be returned for each item. Only the highest bid counts. ie: an user id can only appear at most once in the list. If an user hasn't submitted a bid for the item, no bid is present for that user. A request for a top bids for an item without any bids submitted must be an empty string.

Request: GET /itemID/topBidList

Response: JSON array of {“userID”:“bid”}

<itemID>: 31 bit unsigned integer

<sessionKey>: A session key string retrieved from the login function.

<bid>: double

Example: http://localhost:8081/2/topBidList - >

```
[
      {"23": "33.5"},
      {"467": "32.5"}
]
```