Java Back-End Developer Test

1 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

We acknowledge it will be difficult to write a "complete" and topnotch solution in 2 hours. 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.

2 Description

Write a mini game backend in Java which registers game scores for different users and levels, with the capability to return high score lists per level. There shall also be a simple login system in place (without any authentication...). Deliver a zip file containing:

- The code in the src folder
- A compiled version in a executable .jar file in the root folder
- An optional readme.txt or .pdf with thoughts and considerations around the program

3 Functional requirements

The functions are described in detail below.

- Numbers, parameters and return values are sent in decimal ASCII representation as expected (ie no binary format).
- Users and levels are created "adhoc", the first time they are referenced.

3.1 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”.

3.2 Post a user's score to a level

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

3.3 Get a high score list for a level

Retrieves the high scores for a specific level.

- The result is a comma separated list in descending score order.
- No more than 15 scores are to be returned for each level.
- Only the highest score counts. ie: an user id can only appear at most once in the list.
- If a user hasn't submitted a score for the level, no score is present for that user.
- A request for a high score list of a level without any scores submitted shall be an empty string.

4 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.
- There is no need for persistence to disk, the application shall be able to run for any foreseeable future without crashing anyway.