# Java backend developer test

## 1 Purpose

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

- The code structure and documentation
- Consideration of performance and concurrency issues
- The solution design
- Choice of data structures

We would like to understand your thought process on these main points, so you should comment on whatever you deem necessary for that (choices, trade-offs, etc).

## 2 Description

Write a HTTP-based mini game back-end in Java which registers score points for different users, with the capability to return the current user position and high score list.

Deliver your solution as a github repository with a readme containing instructions to execute.

## 3 Nonfunctional requirements

This should be an in-memory solution for performance reasons, so do not persist to the disk or use a database. Pay attention to performance and concurrency issues that may arise when serving thousands of players.

## 4 Functional requirements

The functions are described in detail below and the notation <value> means a call parameter value or a return value. Numbers parameters and return values are sent in decimal ASCII representation as expected (ie no binary format).

Users are created "ad-hoc", the first time they post a score.

## 4.1 Post a user's score points

This method can be called several times per user and not return anything. The points should be added to the user's current score.

```
Request: POST /score

Request body:

{
   "userId":<userId>,
   "points":<points>
}

Response: (nothing)


<userId> : unsigned integer number

<points> : unsigned integer number
```

## 4.2 Get the current position of a user

Retrieves the current position of a specific user, considering the score for all users. If a user hasn't submitted a score, the response must be empty.

```
Request: GET /<userId>/position

Response:

{
   "userId":<userId>,
   "score":<score>,
   "position":<position>
}


<userId> : unsigned integer number

<score> : unsigned integer number

<position> : unsigned integer number
```

## 4.3 Get a high score list

Retrieves the high scores list, in order, limited to the 20000 higher scores. A request for a high score list without any scores submitted shall be an empty list.

```
Request: GET /highscorelist

Response:

{
   "highscores":[
      {
         "userId":<userId>,
         "score":<score>,
         "position":<position>
      },
      {
         "userId":<userId>,
         "score":<score>,
         "position":<position>
      }
   ]
}

<userId> : unsigned integer number

<score> : unsigned integer number

<position> : unsigned integer number
```