Coding challenge

The Goal

The goal of this test is for us to have some code to chat about on the interview, and for you to showcase your programming skills.

Please note that the test is not so much about finishing and solving the problem, but about delivering a well designed solution with code, tests and documentation that you find of good quality. Because we are mainly working with Java, we would like you to do this in Java.

We are asking you to program a **Java RESTful Web Service** that runs a game of 6-stone Kalah. The general rules of the game are explained on Wikipedia: https://en.wikipedia.org/wiki/Kalah and also below in this document. Please note that the Wikipedia article explains 3 and 4-stone Kalah; we would like your implementation to be 6-stone.

This web service should enable to let 2 human players to play the game, each in his own computer. There is no AI required.

Kalah Rules

Each of the two players has **six pits** in front of him/her. To the right of the six pits, each player has a larger pit, his Kalah or house.
At the start of the game, six stones are put in each pit.

The player who begins picks up all the stones in any of their own pits, and sows the stones on to the right, one in each of the following pits, including his own Kalah. No stones are put in the opponent's' Kalah. If the players last stone lands in his own Kalah, he gets another turn. This can be repeated any number of times before it's the other player's turn.

When the last stone lands in an own empty pit, the player captures this stone and all stones in the opposite pit (the other players' pit) and puts them in his own Kalah.

The game is over as soon as one of the sides run out of stones. The player who still has stones in his/her pits keeps them and puts them in his/hers Kalah. The winner of the game is the player who has the most stones in his Kalah.

Endpoint design specification

1. Creation of the game should be possible with the command:

```
curl --header "Content-Type: application/json" \
    --request POST \
    http://<host>:<port>/games
```

Response:

```
HTTP code: 201
Response Body: { "id": "1234", "uri": "http://<host>:<port>/games/1234" }
```

id: unique identifier of a game
url: link to the game created

2. Make a move:

*curl --header "Content-Type: application/json" *
*--request PUT *
http://<host>:<port>/games/{gameId}/pits/{pitId}

gameId: unique identifier of a game
pitId: id of the pit selected to make a move. Pits are numbered from 1 to 14 where 7 and 14 are the kalah (or house) of each player

Response:

HTTP code: 200
Response Body:
{"id":"1234","url":"http://<host>:<port>/games/1234","status":{"1":"4","2":"4","3":"4","4":"4","5":"4","6":"4","7":"0","8":"4","9":"4","10":"4","11":"4","12":"4","13":"4","14":"0"}}

status: json object key-value, where key is the pitId and value is the number of stones in the pit