Fibonacci Numbers Game

Document version 1.01

Introduction

‘Fibonacci Numbers Game’ or ‘Numbers Game’ is a multi-player turn-base game, in each turn a player selects 3 numbers and these numbers assigned to that player, so other players can’t select these numbers again.

Player score is calculated based on count of Fibonacci numbers assigned to that player. For example if ‘Player1’ started the game by selected {3, 4, 5} then his score will be 2 as (3 and 5 are Fibonacci numbers), then if ‘Player2’ selected {5, 6, 8} then ‘player2’ score will be 1 as (5 is already selected by ‘Player1’ and 8 is Fibonacci) and so on.

Your company assigned to you the task of implementing **REST API** gaming engine server for ‘Numbers Game’ using **Java**, **Spring-boot** technology and store running games related data on **MySQL** database, taking in consideration that restarting the gaming engine server should not impact running games statuses.

Your source code must include **Java Docs** and **Unit Test** cases for each implemented class, and preferred integration testing.

REST API Specifications

Create a Game

Create new ‘Numbers Game’ by providing the player names as list of string with maximum list size of 10 players per a game and player name only English letters and numbers with maximum size of 8 chars.

API Client may create many games with similar or different player names, API Server should accept up to 100 concurrent games.

Returned game code and player codes must be unpredictable random numbers. All game related information must be stored in DB

HTTP Verb `POST`
`URL` `/api/new-game`
Request

```
{
	"players": ["player1", "player2", "player3"]
}
```

Response

```
{
	"game-code":<number>,
	"player-codes":[
				{
					"name":"player1",
					"code":<number>
				},
				{
					"name":"player2",
					"code":<number>
				},
				{
					"name":"player3",
					"code":<number>
				}
		]
}
```

End Game

Stop running ‘Numbers Game’ by its code. All game related information should be removed from DB.

HTTP Verb `POST`

URL `/api/<game-code>/end`

Request

`N/A`

Response

`N/A`

Game Score

Return score for each player. Player score is the count of Fibonacci numbers assigned to that player.

HTTP Verb `GET`

URL `/api/<game-code>/score`

Request

`N/A`

Response

```
{
	"scores":{
		"player1":<number>,
		"player2":<number>,
		"player3":<number>
	}
}
```

Get On-Turn Player

Return the player who has the turn and able to select numbers. Game turns must not exceed 20 turns per each player, after that plays should stop the game.

HTTP Verb `GET`
URL `/api/<game-code>/turn`
Request
`N/A`
Response

```
{
	"next":"player1"
}
```

Play A Move

Player selects only 3 numbers and game engine assign them to him and update the score, considering the new Fibonacci numbers within this game.

HTTP Verb `POST`
URL `/api/<game-code>/<player-code>/play`
Request

```
{
	"Numbers":[3,4,5]
}
```

Response

```
{
	"player-score":<number>
}
```

Error Handling

In case of request data doesn't meet the required game specification, server should return HTTP Code 400 and JSON exception payload '{ "error":"<readable text massage describe the error>"}'

In other server related errors, server should return HTTP Code 500 and JSON exception payload '{ "error":"<readable text massage describe the error>"}'