Contents

[Overview 2](#_Toc129616035)

[Requirements 2](#_Toc129616036)

[Additional Notes 4](#_Toc129616037)

[General Flow 4](#_Toc129616038)

[Criteria 5](#_Toc129616039)

[SQL 6](#_Toc129616040)

## Overview

In this activity, you will write a REST server to facilitate playing a number guessing game known as "Bulls and Cows" / Mastermind /Wordle.

In each game, a 4-digit number is generated where every digit is different.

For each round, the user guesses a number and is told the exact and partial digit matches.

* An exact match occurs when the user guesses the correct digit in the correct position.
* A partial match occurs when the user guesses the correct digit but in the wrong position.
  + These two checks don’t need to be done in the same looping construct.
    - One for loop to check in-line (*CounterInLine++*)
    - Second to check in-position (*CounterInPosn++, CounterInLine--*)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | 3,5,8 | 8 is in Posn so dec In Line |
|  | **P** | **P** | **o** | **E** | In Line | In Posn |
| Guess | 3 | 5 | 7 | 8 | 2 | 1 |
| Line | 1 | 3 | 5 | 8 |  |  |

1358

Once the number is guessed the user wins the game.

*If (InPosn == 4) Won*

## Requirements

You'll create a

1. Spring Boot REST application using
2. JDBC Template to access the database.

A Game should have an answer and a status (in progress or finished). While the game is in progress, users should not be able to see the answer. The answer will be a 4-digit number with no duplicate digits.

int x[] = {0, 0, 0, 0};  
 boolean isDuplicate = false;  
for (int i = 0; i < x.length; i++) {  
 int y = (int) (Math.*random*() \* 9) + 1;  
 for (int j = 0; j < x.length; j++) {  
 if (x[j] == y) {  
 isDuplicate = true; *// Flag used to denote duplicate* break ;  
 }  
 }  
 if (isDuplicate) {  
 i--;  
 isDuplicate = false;  
 } else {  
 x[i] = y;  
 System.*out*.println(x[i]);  
 }  
 }

Each Round will have

1. a guess,
2. the time of the guess (optional!!!)
3. and the result of the guess in the format "e:0:p:0" where "e" stands for exact matches and "p" stands for partial matches.

You will need several REST endpoints for this:

* "begin" - POST – Starts a game, generates an answer, and sets the correct status. Should return a 201 CREATED message as well as the created gameId.
  + Instantiates a game with all the details e.g. Game a = new Game();
* "guess" – POST – Makes a guess by passing the guess and gameId in as JSON.
  + E.g. a.guess(1,4,6,2) … talking to the method guess() which takes in 4 ints
  + The program must calculate the results of the guess and mark the game finished if the guess is correct.
  + ~~It returns the Round object with the results filled in.~~
* "game" – GET – Returns a list of all games. Be sure in-progress games do not display their answer.
  + Select \* from games … A Statement into theTable storing all results from previous games
* "game/{gameId}" - GET – Returns a specific game based on ID. Be sure in-progress games do not display their answer.
  + Select \* from games where ID = ? … A PreparedStatment intoTable storing all results from previous games
* "~~rounds/{gameId} – GET – Returns a list of rounds for the specified game sorted by time~~ – JUST STORE # OF ROUNDS IN GAMES TABLE.

You should include a Service layer to manage the game rules, such as

1. generating initial answers for a game and
2. calculating the results of a guess.

All of your public DAO interface methods should be tested thoroughly.

## Additional Notes

* Use Postman to verify your endpoints behave the way you expect them to be.

## General Flow

1. Start the game
2. Main Menu
   1. Play New Game
   2. List All Games Played Historically
   3. Find Previous Game by ID
3. Playing of Game
   1. Game object (GO) created for that game – Not all fields will be updated yet!
      1. Time of Game (nice to have!)
      2. Answer – The combination in maybe an Array of ints
      3. Status – Boolean?
   2. Computer Generates Combination
      1. Unique values between 1-9 inclusive
   3. You take a guess
      1. EP combination given for that guess
      2. Easy v Difficult
   4. Continue until Game Won or # of Rounds met
4. Once Game Won / Ended (Lost)
   1. Write to DB
   2. Map
5. Listing of all games – Historical record of all COMPLETED games
   1. SQL Query – Select \* … Use a Mapper object
6. Find game by ID

|  |  |  |  |
| --- | --- | --- | --- |
| # | **Criteria** | Meets Spec | Improvement Needed |
| 1 | JdbcTemplate Configuration: JdbcTemplate is properly configured in application.properties and is injected into Daos. | ***5*** | ***3*** |
| 2 | JdbcTemplate CRUD: Data is retrieved, created, updated, and deleted in the database using JdbcTemplate. | ***15*** | ***8*** |
| 3 | Dao tests: Dao tests use a test database. They do not alter the application database. | ***5*** | ***3*** |
| 4 | Java class model: The Java class model represents the correct data and relationships. The query strategy supports the model. | ***5*** | ***0*** |
| 5 | Spring Boot Application: The application runs as a Spring Boot application. Dependencies are injected with annotations. | ***5*** | ***0*** |
| 6 | Service Components: The application uses service components to store and apply game rules. | ***10*** | ***0*** |
| 7 | Testing: Game rules are tested and verified correct. | ***10*** | ***0*** |
| 8 | Spring MVC Controller: The application handles HTTP requests with at least one Spring MVC controller. | ***10*** | ***0*** |
| 9 | Controllers and Daos: Controllers never directly access a Dao. | ***10*** | ***0*** |
| 10 | Controller Handler Methods: Controller handler methods are configured via annotations with correct URL and HTTP method. | ***10*** | ***0*** |
| 11 | Game Play: Can play a full Guess the Number game with Postman. | ***10*** | ***0*** |
| 12 | Code Style: The code style adheres to all Java code conventions and idioms. | ***10*** | ***0*** |

## Criteria

## SQL

DROP DATABASE IF EXISTS BullsCows;

CREATE DATABASE BullsCows;

CREATE TABLE Game(

game\_id INT PRIMARY KEY AUTO\_INCREMENT,

startedTime datetime,

numberOfGuesses int,

answer varchar(4),

isWon BOOLEAN );