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 |
|  |  |  |  |  | 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 generating initial answers for a game and 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.

Criteria

|  |
| --- |
| 1. JdbcTemplate Configuration: JdbcTemplate is properly configured in application.properties and is injected into Daos. |
| 1. JdbcTemplate CRUD: Data is retrieved, created, updated, and deleted in the database using JdbcTemplate. |
| 1. Dao tests: Dao tests use a test database. They do not alter the application database. |
| 1. Java class model: The Java class model represents the correct data and relationships. The query strategy supports the model. |
| 1. Code Review - HTTP methods, URLs, and JSON: During the code review, the apprentice can pinpoint where HTTP methods, URLs, and JSON are used and can explain what they do. |
| 1. Spring Boot Application: The application runs as a Spring Boot application. Dependencies are injected with annotations. |
| 1. Service Components: The application uses service components to store and apply game rules. |
| 1. Testing: Game rules are tested and verified correct. (Some tests) |
| 1. Spring MVC Controller: The application handles HTTP requests with at least one Spring MVC controller. |
| 1. Controllers and Daos: Controllers never directly access a Dao. |
| 1. Controller Handler Methods: Controller handler methods are configured via annotations with correct URL and HTTP method. |
| 1. Game Play: Can play a full Guess the Number game with Postman. |
| 1. Code Style: The code style adheres to all Java code conventions and idioms. |

SQL

DROP DATABASE IF EXISTS BullsCows;

CREATE DATABASE BullsCows;

CREATE TABLE todo(

game\_id INT PRIMARY KEY AUTO\_INCREMENT,

startedTime datetime,

numberOfGuesses int,

answer varchar(4),

isWon BOOLEAN );