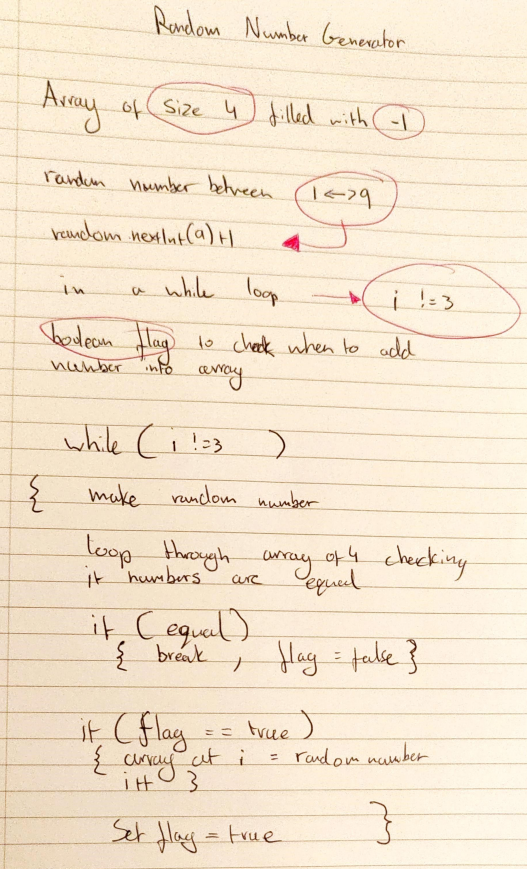
Overview

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

In each game,

1. a 4-digit number is generated
   1. (Random # Generator between 0-9)
2. where every digit is different.
   1. Uniqueness

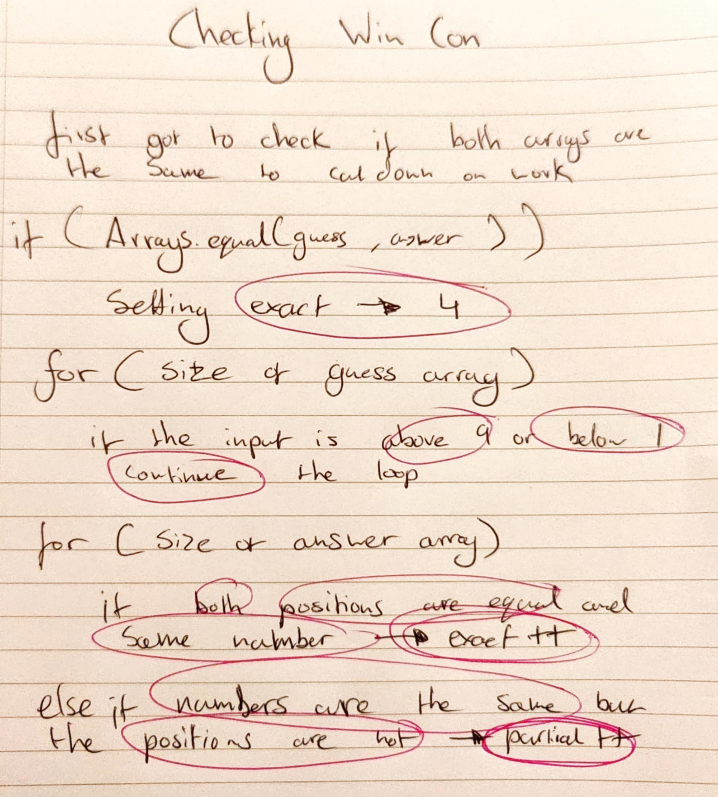


For each round,

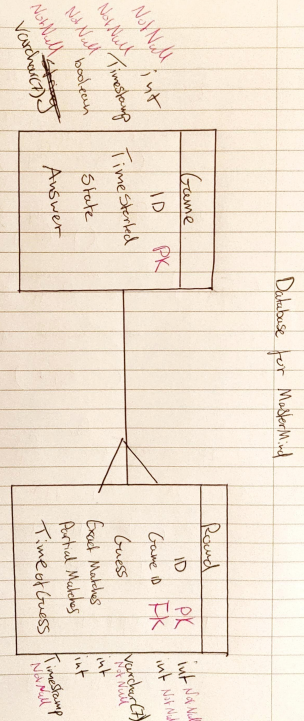
1. the user guesses a sequence of numbers and (4 guesses)
2. is told the exact and partial digit matches (after all four are entered and after each go).

* 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.

Once the number is guessed (exact matches for all digits) the user wins the game.



Requirements

You'll create a Spring Boot REST application using JDBC Template to access the database.

A Game should have an answer and a status (in progress or finished).

Finished – When user gets all 4 correct or just exits

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.

You will need several REST endpoints for this:

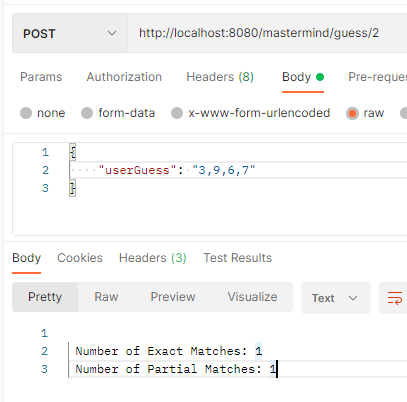
* ~~"begin" - POST –~~ 
  + ~~Starts a game,~~
  + ~~generates an answer,~~
  + ~~and sets the correct status.~~

*Should return a 201 CREATED message as well as the created gameId.*

*Outputs the object that is created while not showing the answer*

**

* ~~"guess" – POST –~~ 
  + ~~Makes a guess by passing the guess and gameId in as JSON.~~
  + ~~The program must calculate the results of the guess~~ 
    - ~~Guess #~~
    - ~~Guess Result (E:P)~~
  + ~~and mark the game finished if the guess is correct.~~

Instead of passing ID in through the body, its passed through the path

* ~~"game" – GET –~~ 
  + ~~Returns a list of all games.~~
  + ~~Be sure in-progress games do not display their answer.~~

All the outputted games that are not finished will display Currently Confidential

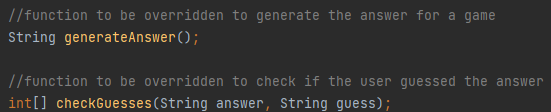
* ~~"game/{gameId}" - GET –~~ 
  + ~~Returns a specific game based on ID.~~
  + ~~Be sure in-progress games do not display their answer.~~

Will output currently confidential if game is in progress



* ~~Guess/{gameId} – GET –~~ 
  + ~~Returns a list of Guesses for the specified game sorted by time.~~

Will be sorted anyway as the oldest will be last

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.

Criteria

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