Java Coding Challenge - Battle of Monsters

Introduction

You will create a screen recording video of yourself completing the challenge, then send me a link to the file via Google Drive. A few things to consider:

- We ask that you complete this challenge within the timeframe agreed on in our conversation.
- **You MUST NOT edit your video, stop it and continue later, copy contents from hidden screens, or anything similar that can be considered cheating. The recording must be without stopping and with no editing.**
- **You cannot use tools such as Copilot, Tabnine, Captain Stack, GPT-Code-Clippy, chatGPT, or similar to simplify or generate code to support the challenge. Doing this will be grounds for automatic disqualification.**
- You can use screen recording software like Loom, QuickTime, or something similar, to create the video.
- The recording should be of the entire coding challenge, from beginning to end, which is about 70 minutes.
  - **You will use 45 minutes for the coding challenge and 25 minutes for the problem-solving challenge.**
  - Remember to record the coding challenge and the problem-solving challenge **together in just one video**; please **do not stop/play and join the video**; if you do it, you will be disqualified.
  - You can spend more than 45 minutes on the coding challenge, but we will discount points on your challenge. But please, be careful with the 25 minutes limit on the problem-solving challenge; you should avoid spending more than the time limit.
- You should record your entire screen so we can validate your implementation correctly. **Also, your computer clock should be visible in the entire video.**
- Please upload the video file to Google Drive and share an open link with us (we support .mp4, files smaller/with less than 4 Gb).
- As you complete the challenge, please explain what you are doing. Walk us through your thinking, explain your decisions, etc. Show us your UI work, if applicable.
- Here is a short clip from a recent coding challenge as an example of what your recording should look like: Example video. It is from a React challenge, but it is the same for any challenge.

The Coding Challenge

Getting Started

Hi! Welcome to the Java coding challenge! We want to check your Java knowledge by giving you an application to finish. We already have an application, but it is incomplete; your goal is to finish this application and make the application usable.

> *We have a project structured with a pattern, so the time suggested above can be done following the pattern and splitting your time into categories; a good organization will lead you to finish the challenge successfully!*

⏳ **Recommended duration for each category in the challenge (this is a suggestion)**
- **Battle Algorithm:** 15 minutes

- **Database**: 10 minutes
- **Data Structure Manipulation**: 10 minutes
- **Unit testing**: 10 minutes

Of course, you can split the time as you want, but we recommend you follow this.

Instructions

The app is a battle of monsters, where we have many different monsters with different characteristics like attack and defense, for example, and we can let them fight each other.

We have implemented almost the entire CRUD for the battle of monsters app, and we have a battle endpoint to list all battles.

Your goal is to implement the missing functionalities, which are the endpoints to list all monsters, start a battle, and delete a battle. Taking a look at all tests will make your path easier.

Remember to implement all the necessary tests to ensure the app works properly. **Remember that the already implemented tests shouldn't be modified.**

You should also take note of the linter and prettier. The linter is currently passing and must pass on completion of the challenge without any modifications to the config.

Also, you will face some issues in making the app run, this is part of the challenge, and we expect you to fix them.

Battle Algorithm

For calculating the battle algorithm, we should take into account the flow below:

- The monster making the first attack is the one with the highest speed; if both speeds are equal, attacks first the monster with the higher attack.
- For calculating the damage, subtract the defense from the attack `(attack - defense)`; the difference is the damage; if the attack is equal to or lower than the defense, the damage is 1.
- Subtract the damage from the HP `(HP = HP - damage)`.
- The battle has many turns until one monster win, and all turns should be calculated in the same request, which means that with only one request to the battle endpoint, you should receive the winner data.
- Who wins the battle is the monster subtracting the enemy's HP to zero.

Technologies

This project is built using the Node ecosystem libraries; it is good you know the following items to have a good performance:

- Java
- Spring Boot
- Gradle
- Flyway
- Project Lombok
- JUnit
- Mockito

Acceptance Criteria

1. Tests were implemented for all new code implementations, and code coverage should be at least 80%, and you must run it and show it to us during the recording.
2. Failing tests already implemented should pass.
3. All monster endpoints were implemented and are working correctly.
4. All battle endpoints were implemented.

Project Setup

Create `JAVA_HOME` and install Gradle in the local environment

```
./gradlew build
```

```
./gradlew build
```

Usage

Run the app:

```
./gradlew bootRun
```

Run the PMD:

Shell Copy

```
./gradlew pmdMain
```

Run the tests:

```
./gradlew test
```

Steps

1. Clone the repository you received by email.
2. Set up the app and get it running. Verify that the linter passes and the test suite fails.
3. Implement tests marked with `@TODO`.
4. Add missing functionalities, so the other tests pass; all implemented tests should not be modified; you only have to create the new ones.

The Problem-Solving Challenge

Hi! Welcome to the Java problem-solving code challenge; we want to test your problem-solving skills here.

You will be provided a repository with some challenges and have to solve them by implementing and making all the created tests pass.

> Please, remember to **NOT USE GOOGLE** to solve the reasoning challenges; try to solve them by yourself.

Challenges

We will provide three challenges on the repository; you will have a time limit mentioned above minutes to solve them and make them pass the tests.

If you cannot finish all challenges, send the record with what you solved, and we will evaluate it.

Remember to solve them using pure Java, and avoid using libraries or anything that is not your code; we want to see you thinking and solving the challenge.

You can do them in any order, but remember that some challenges are more complex than others.

At the end of the challenge, run the tests for us before stopping recording. This step is critical; if you forget to do it, it will cause your disqualification.

Running the Problem-Solving Challenges

The problem-solving challenges are located inside the `co.fullstacklabs.problemsolving` package.

Acceptance Criteria

1. The code should be readable.
2. The code should be easy to maintain.
3. The code should not be complex.
4. All tests should pass.
5. **Tests pass.**