Spartez Coding Test - Event System

During the test you will be improving the code you will receive when the test starts, implementing change requests raised by its users.

The interviewer will ask you to share the screen with your IDE to assess your code reading, code writing, problem solving and communication ability.

All the tasks involve modifying and adding to the code and observing the results.

How to prepare for the coding test

Please take the time to familiarize yourself with this document and the code you received.

Take this time to:

- Read this document carefully.
- Understand the general structure of the project.
- Understand the **contract** of the system described in its API. This is the most crucial step for success, please don't skip it.

The code you received is a simplified version of the project you will work on during the coding test. It is meant to let you familiarise with the project structure and the API.

The code you will receive at the start of the interview is a more advanced version of this project, but it has the same structure, dependencies and API.

How to work during the coding test

The coding test resembles a pair programming session.

The interviewer will be looking at how you use IDE features and work with the code, but also help you spot a missing semicolon, remind the name of the method you are looking for or ask clarifying questions when you get stuck.
It is also perfectly fine if you use Google, Stack Overflow or other help that you find useful.

When the test starts you become the owner of the code - you may change it however you see fit. Mind the time limits, though - your success depends on finishing the tasks.

We believe the most effective approach to the coding test is:

- Make sure you understand the task before starting on a solution. Ask for clarification of any aspect of the task you are hazy about. Explain your understanding of the task to the interviewer - it's an early opportunity to correct any misunderstandings.
- Explain what you are about to do in the code. This demonstrates your professionalism but also gives opportunity to validate your approach.
- Ask for help on pesky details. If you can't remember a method name you want to use, we can prompt you. We do not expect you to have the Java API memorized.
- If there is something you would do with the code in a real project but you feel you don't have enough time for that, please describe it to the interviewer!

Project you will work on

During the test you become a maintainer of a simplified implementation of an event system.

Event system serves as intermediary between components that create Events and publish them using the Event Manager and components that register possibly multiple Event Listeners to receive notification about those Events.

The implementation is simplified to serve coding test needs, it misses details that would be required in a real production library. However, in contrast to the code you just received, it is complete enough to serve its primary purpose.

The code

Event Manager is specified by the **EventManager** interface and implemented by the DefaultEventManager class. Listeners are registered in the EventManager and must implement the **EventListener** interface.

When EventManager is called to publish an **Event,** it uses methods on all registered EventListeners to determine if the listener is interested in a specific event and if so, the handleEvent() method is called on such listener with the triggering event as the parameter.

The API of the system is documented in the JavaDoc comments of the mentioned interfaces. They have been very carefully written and contain crucial information about the expected behaviour of the code. You will need this information to implement your tasks.

Project Structure

The code follows the standard Maven structure:

- production code is in the `src/main/java` folder,
- tests are in `src/test/java` folder,
- project file to be opened in your IDE is the `pom.xml`