Strands Interview coding test

Disclaimer

Table of Contents

Strands Java Test for Interview

About the Test

This coding test is comprised of a series of problems of increasing difficulty designed to enable the interviewer to assess a candidate's code reading, code writing and problem solving ability.

The code used for the test is a mini event system. Event systems like this are a common way of maintaining a loose coupling between components in a larger system. If a component needs to perform some action as a consequence of an action of some other component, it registers to receive relevant event types. Components then use the Event Manager to re specific events and all registered subscribers wishing to receive events of that type (dependent components) are notified.

The coupling is loose because it is indirect, mediated by the publish-and-subscribe facility of the Event Manager. This is the central idea behind Message-Oriented Middleware (MOM).

The benefits of loose coupling are that expansion and reconfiguration of the system results in minimal knock-on effects to components that re or receive events. New listeners and event types should be able to be added with minimal disruption to existing code. Most tasks in this test focus on making those sort of changes to this system.

Key classes include the EventManager interface and its main implementation DefaultEventManager. EventListener implementations register with EventManager to receive events. When the EventManager is told to re an event, all listeners are consulted to see if the event is one that they should handle. The methods on the EventListener interface are used by the EventManager to determine if the listener is interested in a specific event and if so, to "handle" or receive that event.

The code is in two source directories with identical package structure, "main" and "test". Production code belongs in "main" and JUnit unit tests belong in "test". The convention is that a class in "main" called Xyz will be tested by a class in "test" called XyzTest and will be in the same package. Not all production classes have a corresponding Test class.

The code is not intended to present as perfectly written and bug-free. In fact the less you assume about it and the more you rely on your own reading of it, the better able you will be to perform the problem-solving tasks.

Challenge

To complete this challenge you must do the exercise in less a hour, you should create a new local repository in your working directory:

```
git init
```

There you will commit your code in different phases. When you have created the repository, you must create a new “master” branch, you will only use this branch in the challenge.

Add the source code we have send you attached to the mail to your repository and you commit these changes using the message “First commit” and push in the public repository.

You must send us an email with the ZIP with the local repository.

Task 1: create a new unit test (20 minutes)

Focus on SimpleEvent and SubEvent classes. SubEvent class extends to SimpleEvent class, but in this event manager version when we send a new event using SubEvent class, the listeners of SimpleEvent don’t receive notification. Our task is create a new test that it verifies this feature of our event manager. When you finish commit and push your code with the message “Task 1”.

Task 2: listen everything (20 minutes)

We want to add new feature to our event manager, we want you to modify the event manager to add a new special listener.

When a new event listener is added, if it returns an empty array when the “getHandledEvent-Classes” method was called, this event listener must listen all events in the system.

You must add this feature trying to reduce the impact in the code, you don't need to add a new let to the repository.

To finish this task, you must include a test to verify this new feature and you must commit and push your code with the message “Task 2”.

Task 3: parents and children (20 minutes)

On the task 1, we check that our code don’t support aggregate listeners, if you create a listener by a super class, you won’t receive event for a subclasses.

Now we want to change this behaviour. Modify the code to support this new feature and change the test with the new specification.

To finish this task, you must commit and push your code with the message “Task 3”.

(OPTIONAL): Extra mile

How was the warmup? Easy ;). Don’t you thing so?. From this point, the tasks described below are optional. If you steel feel you want to challenge yourself please go ahead. For the following block, we think you should deserve an extra hour.

Task 4: Deeper Understanding (17 minutes)

Create a class that only can have one instance running at the same time (in the same VM). The first thread fetching an instance of this class should result in a instance of this class being created. Further threads fetching an instance of the thread should result in the already created instance to be returned.

Task 5: Databases and SQL (30 minutes)

The following tables are given in the database: User, Account, Transaction, Category. (a user can have several accounts, an account has several transactions and each transaction has a system category).

USER – ID, NAME, PK (ID)
ACCOUNT- ID, USER_ID, NAME, UPDATED_DATE, PK (ID)
TRANSACTION - ID, ACCOUNT_ID, CATEGORY_ID, NAME, AMOUNT, POSTED_DATE, PK (ID)
CATEGORY – ID, NAME, PK (ID)

For the sake of simplicity, You do not need to worry about other indexes. How would make a SQL query to get the following results?

a) Get the total number of transactions in 2013 of a specific category ('15') and user ('356789').
b) Get transactions of a user filtering by account and current month.
c) Get income categories of a user for the current month. A category is considered INCOME when the amount of the transaction is positive.

Task 6: Regular expressions (15 minutes)

a) Write a regular expression that matches the following email addresses:
f.soler@strands.com / i.tarradellas@strands.com / a.dereina@strands.com

b) Write a regular expression that matches the following string, and is able to extract the date:
ELCORTEINGLES28/5/13
CARREFOUR2/10/13