Merchant Solutions – Technical Exercise

Task

Please review the below and build a solution meeting the specified requirements.

Instructions

Please submit your code to a github account and respond with a link to your code within 3 working days from the date of provision.

Please keep in mind.....

There are no right or wrong answers to this problem. When building your solution work incrementally and describe how each commit has changed the system in the commit message.

You will be assessed on three things:

- Your ability to understand what is being requested
- The merits of the solutions you propose
- The process by which you attempt to solve the problem posed
    - Demonstrate how you incrementally build your code and use your commits to explain their development process and thoughts.
    - Demonstrate that your code works through appropriate testing

If you can't do it all, then please give it a go and do the best you can. We review all submissions.

Oh and Finally.....

Happy Coding!

Requirements

Your team has been asked to write a new Trading application. You work with an Algo team who provide you a library containing all the required Trading Algo software and the job of your platform is to execute the Algo when given a ‘signal’.

In the following section the ‘Algo’ class belongs to the Trading Algo library and **cannot be modified**.

Your team has written the ‘TradingApplication’ code to be able to process one of these signals (a simple integer). Each signal specification is given to your team in the form of a JIRA from your analysts. The role of your team is to implement the new signal specifications and release to production as quickly as possible.

While the current ‘TradingApplication’ code only has three signals, once in production it is expected that up to 50 new signals will be added per month (600 after year one, 1200 after year two etc).

The Task

We would like you to make any changes to your teams code to make the code:

- Receive signals via HTTP instead of the SignalHandler interface. The code should be a running service with a single http endpoint for receiving the ‘signal’ to be processed by your application.
- Easier for your team to understand and debug
- Easier for your team to maintain and add new signals
- Easier to test
    - The code should have appropriate levels of testing to ensure that the stated requirements are met.

```java
/**
 * This is a callback interface from our trading system, and we cannot change it.
 * We want to replace the callback with an HTTP API.
 */
public interface SignalHandler {
    void handleSignal(int signal);
}


/**
 * This is implemented in a third-party library and we cannot change it.
 * The println calls are placeholders for the actual algorithms implemented by the library.
 */
public class Algo {
    public void doAlgo() {
        System.out.println("doAlgo");
    }

    public void cancelTrades() {
        System.out.println("cancelTrades");
    }

    public void reverse() {
        System.out.println("reverse");
    }

    public void submitToMarket() {
        System.out.println("submitToMarket");
    }

    public void performCalc() {
        System.out.println("performCalc");
    }

    public void setUp() {
        System.out.println("setUp");
    }

    public void setAlgoParam(int param, int value) {
        System.out.println("setAlgoParam " + param + "," + value);
    }
}
```

```
/**
* This is your team's code and should be changed as you see fit.
*/
public class Application implements SignalHandler {
    public void handleSignal(int signal) {
        Algo algo = new Algo();

        switch (signal) {
            case 1:
                algo.setUp();
                algo.setAlgoParam(1,60);
                algo.performCalc();
                algo.submitToMarket();
                break;

            case 2:
                algo.reverse();
                algo.setAlgoParam(1,80);
                algo.submitToMarket();
                break;

            case 3:
                algo.setAlgoParam(1,90);
                algo.setAlgoParam(2,15);
                algo.performCalc();
                algo.submitToMarket();
                break;

            default:
                algo.cancelTrades();
                break;
        }

        algo.doAlgo();
    }
}
```