**TRADING APPLICATION**

1. Algo Team Library Files:

To promote code reusability, I have made our Algo team's library files **(*Algo*, *SignalHandler*)** public and organized them in a dedicated package. By doing so, other classes from different packages can now easily access and utilize these libraries.

1. Signal Configuration File (***signal-config.json***):

I have introduced a configuration file named "***signal-config.json***" to manage signal IDs and their corresponding method names. This approach enables us to add new signals without modifying the codebase. In the future, if we deploy the application in **Kubernetes**, we can make this file configurable through the ***ConfigMap*** service. This way, updates to the signal configuration can be made without the need to restart the application.

1. Our ***Application*** Class Renamed to ***SignalProcessor***:

To enhance code clarity and maintain consistency, I have renamed the main application class to "***SignalProcessor***." This name better reflects the class's responsibility of processing trading signals and aligns with its specific role.

1. **AlgoMethodInvoker** **Class**:

The "***AlgoMethodInvoker***" class has been introduced, leveraging Java reflection to dynamically invoke methods from the Algo class based on the provided parameters. This utility class simplifies the process of calling methods with varying argument lists, improving code flexibility and maintainability.

1. **Test Cases** :

Test cases have been designed to evaluate the following scenarios:

1. Valid Signal Handling: Tests to verify that the **SignalProcessor** correctly processes valid trading signals and invokes the appropriate actions on the Algo object.

2. Invalid Signal Handling: Tests to ensure that the **SignalProcessor** gracefully handles invalid or non-existent trading signals without causing any unexpected errors or disruptions.

3. Signal Configurations: Test cases that validate the signal configuration file ("signal-config.json"). These tests ensure that signals are correctly defined, and their corresponding methods are accurately mapped.

4. Method Invocation: Tests to confirm that the AlgoMethodInvoker effectively invokes methods on the Algo object with various types of parameters, including no arguments, and multiple arguments.

5. Error Handling: Test scenarios that assess the application's error-handling capabilities. These tests validate that appropriate exceptions are thrown and logged whenever errors or exceptions occur during method invocation or signal processing.



