Coding Test

Duration: ~1 hour

# Background

The goal of this test is to develop a simple Java program comparing data coming from a provider and reacting to given conditions.

The provider notifies two types of data:

* Market Data: Quantity and Price of the latest exchange transaction on a product
* Pricing Data: Fair Value of a product, calculated by an internal system

When you run the provided Test class, the registered Listeners (*MarketDataListener* and *PricingDataListener*) will start receiving data.

# Functionality to implement

## First Part: Store the Fair Value

Using the provided *VwapTrigger* class, as a beginning, implement the *fairValueChanged* method to store that latest fair value received for each product.

Example

1. fairValueChanged(“P1”, 11.0)
   1. => Fair Value for P1 is 11.0
2. fairValueChanged(“P1”, 10.9)
   1. => Fair Value for P1 is 10.9
3. fairValueChanged(“P2”, 12.5)
   1. => Fair Value for P1 is unchanged
   2. => Fair Value for P2 is 12.5
4. …..

## 

## Second part: Calculate the VWAP

Using the provided *VwapTrigger* class, as a beginning, implement the *transactionOccured* method to calculate the Volume Weighted Average Price (VWAP) over the 5 last transactions received for each product.

If less than 5 transactions were reported on a given product, the VWAP will be calculated using all the transactions available for this product.

Example:

1. transactionOccured(“P1”, 1000, 10.0)
   1. => VWAP for P1 = 1000 \* 10.0 / 1000 = 10.0
2. transactionOccured(“P1”, 2000, 11.0)
   1. => VWAP for P1 = (1000 \* 10.0 + 2000 \* 11.0) / (1000 + 2000) = 10.6666666
3. transactionOccured(“P2”, 500, 12.0)
   1. => VWAP for P1 is unchanged
   2. => VWAP for P2 = (500 \* 12.0) / (500) = 12.0
4. …

## Third Part: Compare the values

Each time a new transaction occurs or a new fair value is provided, compare the latest VWAP with the latest Fair Value.

If the VWAP is greater than the Fair Value, print a log:

VWAP(xxxxx) > FairValue(yyyyy)

with the corresponding VWAP and Fair values.

# 

# Thread-Safety

It’s very important to note that the *DataProvider* uses several threads to notify the data. As a result, the program must be thread safe.

# Expected results

The expected result is a running Java program.

The program will have to check the conditions mentioned above for all the products its listeners receive.

In order to verify that the calculations are correct, using log4j log, the following log in the console and a file is expected for the product having productId = “TEST\_PRODUCT” (each line corresponds to a point where the conditions were met):

VWAP(100.04761904761905) > FairValue(100.0)

VWAP(101.09433962264151) > FairValue(101.0)

VWAP(101.5909090909091) > FairValue(101.5)

VWAP(101.5909090909091) > FairValue(101.0)

# **Material Provided**

* This document
* This Gradle project
* Javadoc of the *DataProvider* related classes and interfaces: *doc/index.html*
* A *resources/log4j.xml* configuration file with a specific “TEST\_PRODUCT” logger in DEBUG mode (this configuration can be changed)
* You can add new code / class / interface etc., act as any production code
* You can use third-party libraries
* You can use any JDK version
* Any IDE or text editor can be used
* Internet access is allowed