MS Technology Analyst Program

Java OO Assessment

This assessment takes the form of a formally reviewed lab exercise. Please complete the exercise on your own. Your instructor will provide details of submission and marking.

1

The Bank of Montreal is looking to trade different types of products. All products have a unique identifying string, known as the product ID. All products also have a value (current price), but it is impossible to calculate the value of a product without knowing what kind of product it is.

The Bank of Montreal is looking to trade 2 kinds of products initially:

* Stocks - Stocks have a stock ticker and an exchange
* Futures - Futures have an exchange, contract code, a contract month and contract year.

For valuing a product a vendor interface is supplied as follows:

public interface ProductPricingService {

double price(String exchange, String ticker);

double price(String exchange, String contractCode, int month, int year);

}

Develop the above domain model, including valuation for use in the following service. **Note that where there are dependencies on the ProductPricingService in your tests you should make use of an appropriate Mocking framework to satisfy the depenencies and you should not attempt to implement the interface manually.**

The following service should be implemented and fully tested. Think carefully about the data structures you will use and how you will test your system.

/\*\*

\* Interface used to audit products that have been registered

\* for tracking. Products can be registered against the

\* API and trades and quantity are tracked. The API is

\* capable of retrieving the total quantity traded for the day

\* and the total value of all the trades.

\*/

public interface MontrealTradedProducts {

/\*\*

\* Adds a new product to the system that

\* the class will track statistics for

\* @param product add a product available for trading

\* @throws ProductAlreadyRegisteredException thrown

\* when a product is registered twice

\*/

void addNewProduct(Product product) throws ProductAlreadyRegisteredException;

/\*\*

\* Books a quantity against the product traded. If the product

\* has not been registered, no quantity is recorded as

\* it is not a product we are required to track.

\* @param product the product traded

\* @param quantity the quantity traded

\*/

void trade(Product product, int quantity);

/\*\*

\* Calculates the total quantity of all the registered

\* traded products so far today

\* @return the total quantity traded

\*/

int totalTradeQuantityForDay();

/\*\*

\* Calculates the total value of all the registered traded products

\* so far today. This is done by multiplying the value by the quantity

\* traded.

\*

\* @return the total value of today's traded products that are

\* registered in the system

\*/

double totalValueOfDaysTradedProducts();

}