#pragma once

#include <string>

#include <vector>

class Order

{

public:

// do not alter signature of this constructor

Order(const std::string& ordId, const std::string& secId, const std::string& side, const unsigned int qty, const std::string& user,

const std::string& company)

: m\_orderId(ordId), m\_securityId(secId), m\_side(side), m\_qty(qty), m\_user(user), m\_company(company) { }

// do not alter these accessor methods

std::string orderId() const { return m\_orderId; }

std::string securityId() const { return m\_securityId; }

std::string side() const { return m\_side; }

std::string user() const { return m\_user; }

std::string company() const { return m\_company; }

unsigned int qty() const { return m\_qty; }

private:

// use the below to hold the order data

// do not remove the these member variables

std::string m\_orderId; // unique order id

std::string m\_securityId; // security identifier

std::string m\_side; // side of the order, eg Buy or Sell

unsigned int m\_qty; // qty for this order

std::string m\_user; // user name who owns this order

std::string m\_company; // company for user

};

// Provide an implementation for the OrderCacheInterface interface class.

// Your implementation class should hold all relevant data structures you think

// are needed.

class OrderCacheInterface

{

public:

// implememnt the 6 methods below, do not alter signatures

// add order to the cache

virtual void addOrder(Order order) = 0;

// remove order with this unique order id from the cache

virtual void cancelOrder(const std::string& orderId) = 0;

// remove all orders in the cache for this user

virtual void cancelOrdersForUser(const std::string& user) = 0;

// remove all orders in the cache for this security with qty >= minQty

virtual void cancelOrdersForSecIdWithMinimumQty(const std::string& securityId, unsigned int minQty) = 0;

// return the total qty that can match for the security id

virtual unsigned int getMatchingSizeForSecurity(const std::string& securityId) = 0;

// return all orders in cache in a vector

virtual std::vector<Order> getAllOrders() const = 0;

};