ESS 201 Programming II
T1 2022-23
Assignment 6/Mini-project
C++ and Java
Version 1.1

v1.0: Description of assignment
v1.1: Details of file formats for messages between platform and portal, and definition of base classes

Overview
In this assignment, we will simulate a simple e-commerce site. This site consists of 3 sets of players: Platform, Sellers, Customers. Sellers offer Products through the Platform, which are viewed by Customers, who may then decide to order some of the products.

Customers interact with the system using the Customer Portal. Sellers attach themselves to the platform, while the Portal is also connected to the platform. Thus all interactions between customers and sellers are through the platform.

The Portal allows customers to view the list of available product categories. For each category, they can ask for a listing of available products. On viewing the listing (which can be sorted as per their preference), they may choose to purchase one or more units of a given product. Once sold, the product is no longer available to others.

When a customer makes a request (either for listing products or buying products), the portal relays the request to the platform. The platform, in turn, sends the request to the relevant sellers, and sends their response back to the portal.

Design approach
Needless to say, we model these as classes. Broadly, we have the following classes (at a minimum):

- Platform: Provides a way for sellers to attach themselves to the platform. Receives requests from the portal (see below), and passes those on to the registered sellers. Gets their responses, and sends the consolidated response to the portal
- Seller: Maintain different categories of products, and different products within each category. A seller can decide which products to offer, and the price for each item. Price can change dynamically, from call to call. Seller also maintains the count of each product and can only sell that many of a given product. . Each seller also assigns a unique name for each product (for instance, it could be <seller name>-<product name>.
- Portal: Mimics the front-end of an application, though here it would be command line based. Users can enter specified commands, which are passed to the platform. On return, the portal sorts the list of products as per the preference of the user and displays them

- Product: provides a common view of all products. You can query the object for its category, unique name, price, and number available
- Subtypes of Product: For our purposes, we have two product categories: Book and Mobile. These are derived classes of Product, and add methods relevant to them.

Implementation

This project is to be implemented partly in Java and partly in C++: The Platform and Sellers in one language, and the Portal in the other. You can choose either language for either set of classes, so long as you use both.

For our implementation, to simplify connection between the platform and portal, we use files to enable communication. The portal writes requests to a file which the platform reads, and responses are written by the platform to a different file that the portal reads. For convenience, we will call these PortalToPlatform.txt and PlatformToPortal.txt. Since responses may come back after arbitrary delays, the portal will periodically check the file for responses. Similarly, the platform periodically checks its input file for new commands.
Note: there are many ways of achieving these kinds of asynchronous interactions. For simplicity, we use “polling”, and we let the user/operator decide when to check for messages/responses. For instance, the user can type “check” in the portal to make the portal check for responses from the platform. Similarly for the Platform end.

Within each team, each member should implement a different seller - that is, a derived class of Seller, where they work out a way to manage the list of products, quantities, pricing etc.Of course, team members collaborate and implement one version of the other classes

Program execution

We run the main of the Java and C++ implementations - one the Platform and the other the Portal - in different windows. The user types in requests in the Portal, and sees the responses shown back in the same window. At the back, the Portal writes out requests as text messages in the file PortalToPlatform.txt. Each request line has a unique id (say an integer). The platform, when writing back the response in PlatformToPortal.txt, adds this request ID to the response details.

The user can type any of the following commands:

- List Category SortOrder
    - shows list of products belonging to Category from all sellers who have that Category, sorted by SortOrder, which is one of Price or Name
    - For each line, it shows:
      Product name, productID, price, number available, and optionally other information. Note that all these values are decided by each seller for the products they offer. ProductID should be unique across all sellers and products.
- Buy productID numItems

    - wants to buy that many items of the specified productID. The seller should ideally reduce their availability count. Also, you cannot buy more items than were offered, Such attempts will not be successful.
- Check
    - Asks the portal to check if there is a response from the platform for an earlier request.
- Note that the above commands can be called in any order. For instance there could be multiple “Buy” commands after a “List” and without any “Check” in between.

The syntax of the requests and responses in the two files are standardized (see FileFormat.txt), so that the platform of one team can interact with the portal of another team (for now running on the same computer!)

For the Platform program, there is only one user input (e.g. Check) that is used to make the platform check for input requests.

Details of abstract classes and file format are in the attached files

Package ecomm contains the base classes. **These files are common for everyone and should not be modified**
Base classes: Platform, Portal, Seller, Product
Globals: constants and data common to all classes

Package demo
contains outline of derived classes that should be implemented.
One derived class of Portal and Platform for each team
Each student of the team should have a sub-directory (sub-package) of demo with a derived class of Seller. This should also contain sub-types of Product that you should implement for the Book and Mobile categories.

Package default
contains sample main files. These should be modified to instantiate the appropriate Platform, Portal and Seller derived classes from package demo (or sub-packages) as needed.

FileFormat.txt provides the syntax of requests and responses to be used in the files PlatformToPortal.txt and PortalToPlatform.txt