Lab session 4: Interfaces
24292-Object Oriented Programming

1 Introduction

In this lab session you have to implement the design of Seminar 4 that models an online store. The aim is to achieve a program that is more complete and has some added features compared to Lab Session 3. The session is mandatory and you have to deliver the source code of the Java project and a report describing the implementation decision and details. We will start by creating a class called FullOnlineStore.

2 The Online Store

The static main method remains in the *OnlineStore* class and plays the role of the main program. In addition of previous Lab sessions, the *OnlineStore* is in charge of instantiating a linked list of sales, and add a new *Sale* every time an item is sold.

The relations of the class diagram are the ones in Figure 1. The modifications of the methods from Lab 3 are open to you but always following the diagram constraints.

3 Taxable Interface

The classes Package and Item should now implement *Taxable* interface because both have a cost to the OnlineStore, so we have to pay taxes in both cases. It contains an static final attribute with the current percentage of IVA to apply to taxable objects.

The methods that define this interface are:

- double getPrice();
- double getPriceOnlyTax();
- double getPricePlusTax();
- double sumTotalTax(Taxable t);

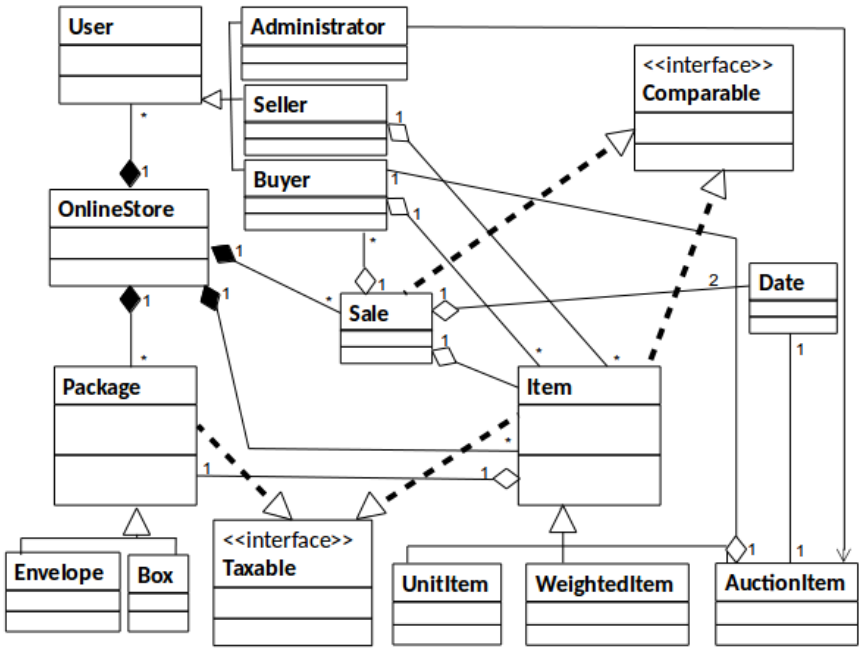

Figure 1: OnlineStore Class Diagram

The idea is to calculate the prices before and after taxes, and generate a method sumTotalTax where we can accumulate a total tax adding all taxes from taxable objects.

4 Managing the Sales of the Store

To represent dates you can create a new class or use the Date class from the Java library *java.util*. To find out how to use the class, read through the documentation:

`https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Date.html`

Make appropriate changes to the class *AuctionItem*. Create the *Sale* class and add the attributes inferred from the class diagram. Also add appropriate methods. Once the *Sale* class has been implemented, add an attribute to *OnlineStore* that models the relation between *OnlineStore* and *Sale* (i.e. the store should contain a register of all past sales).

Finally, implement a method *sell* in the class *OnlineStore*. This method should have several effects:

1. Update the total price and benefit of past sales.

2. Call the method buy of *Buyer* (and possibly *sell* of *Seller*).

3. Create an instance of *Sale* and store this instance in the register of sales.

4. Remove the item sold from the list of items.

4.1 Updating the current sale

To manage auction items and register past sales, we will include an instance of *Date* in *OnlineStore*, representing the current date. Include a method in *OnlineStore* that increments the current date. This method should check if there are auction items that expire on the new date, and if so, perform the sale of these items.

4.2 Sorting items and sales

The online store wants to sort the items by price and the sales by date. To do so, the classes *Item* and *Sale* have to implement the interface *Comparable* (which is an existing interface in the Java library *java.lang*, i.e. does not have to be imported). To do so, these two classes have to implement the lone abstract method *compareTo* of the *Comparable* interface.

In the extended example of vectors (posted in the Aula Global) you will find an implementation of the method compareTo (in the class *MyClass*). Implementing *compareTo* in the classes *Item* and Sale should follow a similar pattern.

Once the classes *Item* and *Sale* implement the *Comparable* interface, you can sort lists of items and sales using the static method sort of the class *Collections* from the *java.util* library.

5 Submission

As usual, in the main method you should create instances of the various classes, and test the different methods to make sure that they work as expected. In particular, there should be a single instance of the class *OnlineStore*, representing the store itself. Create several users and items and add them to the store, and simulate the fact that multiple items are sold using the method sell of OnlineStore.

The deadline for this lab session 4 is before lab session 5. Do not forget to write a report explaining your code, ideas, problems and how you solved them.