Pau Cobacho, Arnau Adan 2 desembre de 2020
POO

Lab session 4: Interfaces

FullOnlineStore

INTRODUCCIÓ DEL PROBLEMA

L'objectiu per aquesta sessió de laboratori és complementar l'aplicació de Java dissenyada per la Online Store del treball passat implementant noves característiques. L'objectiu és doncs aconseguir un programa més complet amb noves funcions.

Se'ns demana dotar a la tenda per:

- portar un registre de les vendes per a la comptabilitat.
- implementar les dates de manera més útil
- aplicar un impost als objectes que es puguin vendre i siguin imposables
- ordenar les vendes segons la data i els productes segons el benefici

El disseny vist pel seminari 4 és el que ens indicarà la estructura adient per satisfer aquests punts. Se'ns proposa atacar la solució creant una interfície anomenada **Taxable** que descriurà el comportament a seguir per tots els objectes imposables de la tenda i l'hauràn d'implementar llavors els objectes de la classe Item i Package, o sigui els productes i paquets. On aquests segons, a diferència del laboratori anterior, ara, tindràn un preu assignat. Implementant aquesta interfície de manera correcta tindrem llavors solucionat el requeriment d'aplicar l'impost.

En quant a millorar el maneig de les dates de la tenda se'ns proposa representar-les implementant una classe d'objectes del tipus data o bé usar alguna classe ja definida per alguna llibreria de Java. Farem els canvis corresponents doncs a les classes que necessitin l'ús d'aquest tipus d'objecte.

Ara, per tal de representar les vendes de la tenda, és a dir; definir les seves característiques i comportaments, crearem una nova classe anomenada **Sale**. Que permetrà a la tenda portar ordenadament un registre sobre aquestes. Si més tard volem ordenar-les per la seva data haurem d'implementar un mètode comparador que faci possible ordenar-les segons desitjem. Farem el mateix a l'hora d'ordenar els productes per preu. O sigui, definirem un nou mètode comparador per ordenar-los com vulguem però implementat a la classe d'Item. Veurem més tard que aquest mètode, que caldrà sobreescriure, vindrà donat per la interfície **Comparable**.

Finalment, per millorar la flexibilitat del programa se'ns demana implementar diversos mètodes nous per manegar les que, a continuació anunciem, diferents accions de la tenda: actualitzar el preu i benefici de la tenda segons les darreres vendes produïdes, comprar i vendre productes, definir noves vendes, esborrar els items ja venuts del catàleg, osbervar si existeixen items pels quals la data límit de subhasta caduca el mateix dia i adjudicar al comprador i ordenar tant vendes com items.

Ja tenim doncs la idea principal sobre el disseny de la solució.

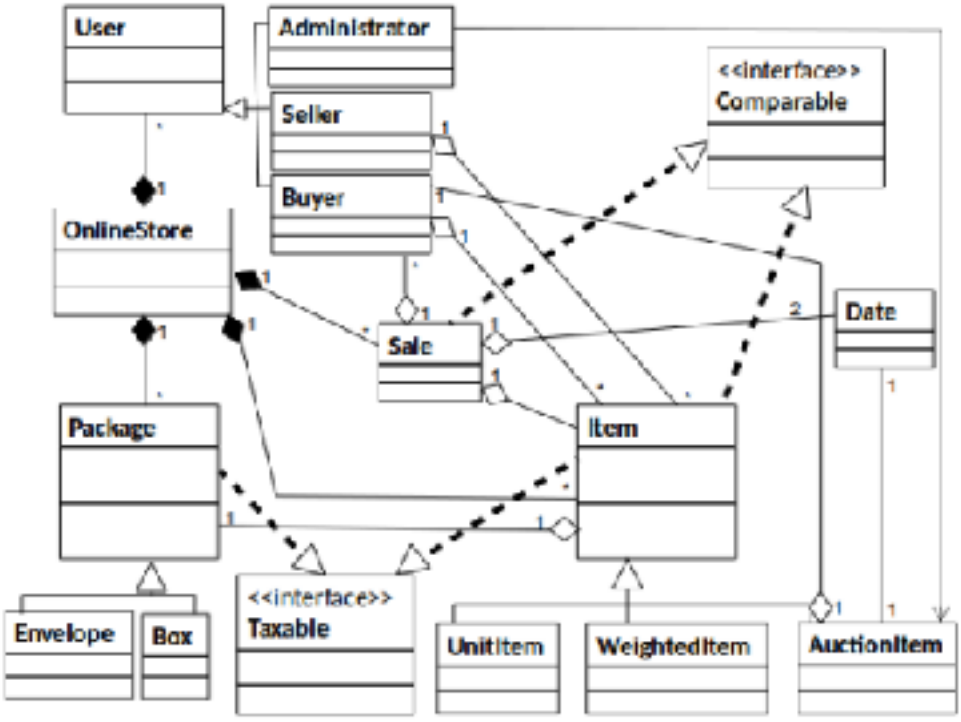

Figura 1: diagrama de classes pel programa OnlineStore.

IMPLEMENTACIÓ DE LA SOLUCIÓ

INTERFÍCIE TAXABLE

Decidim començar implementant la interfície anomenada **Taxable** per dotar als objectes que vulguin usar-la dels comportaments necessaris per manegar l'impost de l'iva, en aquest cas, que hauràn de respectar tant els productes com els paquets.

Definirem el tant per cent de l'impost. Que prendrà forma d'atribut estàtic i final, per tant el mateix per tots els objectes que implementin la interfície amb el valor de l'IVA (21%).

Necessitarem que tots els objectes imposables suguin capaços d'obtenir el seu preu amb i sense impost implementant **getPrice( )** i **getPricePlusTax( )**, obtenir només l'afectació de l'impost en l'objecte actual definint **getPriceOnlyTax( )** i, finalment, generar una suma on es pugui acumular l'impost total afegint tots els impostos dels objectes imposables.

INTERFÍCIE CALENDAR

Un cop solucionat el punt dels impostos continuem per representar les dates més eficientment. Usarem, simplement, la classe ja definida a Java anomenada **Calendar.** Un cop entesa i tenim clars els mètodes que brinda passem a fer els canvis adients a les classes que necessitin informació de la data actual (un canvi és, per exemple, modificar l'atribut d'un item de subhasta sobre el fi de la mateixa canviant el tipus de dada per una del caire Calendar en comptes d'String). AuctionItem i Sale doncs usaràn en potència aquesta interfície, on necessitarem per ambdos casos mantenir una referència sobre la data de cada objecte, com ara la data de venta, d'enviament o caducitat d'una subhasta. També la OnlineStore usarà la data per mantenir informació sobre el dia actual. Resumint; allà on usem la data en format String ara ho farem en format Calendar.

CLASSE SALES

Ara ja ho tenim tot disposat per representar una venda; llavors definirem la nova classe **Sale** creant atributs i mètodes per aquests objectes. Sabem que cada venda serà definida per:

el seu item **saleItem**, el seu usuari comprador **saleBuyer**, el seu paquet d'enviament **salePackage** i les seves dates de venda i enviament **saleDate** i **shippingDate** respectivament.

En quant al comportament d'aquesta classe només ens caldrà definir, encara que també hem imlpementat getters i setters tot i que no s'usen, com voldrem ordenar aquestes vendes. Voldrem fer-ho segons la data, per tant, implementem el mètode **compareTo( )** donat per la interfície ja definida a Java **Comarable**, que estudiem a continuació.

INTERFÍCIE COMPARABLE

Aquesta interfície ens permetrà més tard ordenar els objectes desitjats, items i vendes, amb el procediment que volguem implementant el mètode de la mateixa classe anomenat **compareTo( ).** Sobreescrivint-lo podrem comparar dos objectes del mateix tipus, items per data i vendes per preu. Llavors implementarem aquest mètode tant a Sale com a Item per més tard ordenar els objectes de cada tipus des de la classe principal.

MILLORAR FLEXIBILITAT

Per tal de millorar el funcionament i ordenar la classe OnlineStore hem comentat abans que definiriam certs mètodes per la seva classe.

Definirem el mètode **sell( )** a la classe principal que ens permetrà, donat un usuari comprador, un venedor i un item; acualitzar el preu i benefici actual de la tenda, comprar i vendre el producte per cert comptrador i venedor respectivament, crear una venda amb les dades del producte venut i l'usuari implicat i esborrar tots els productes venuts del catàleg.

Definirem també el mètode **currentDate(** Administrador Admin **)** que necessita un administtrador per investigar si existeixen items de subhasta pels quals la data de finalització

de pujes sigui el dia actual (en cas afirmatiu procedeix a realitzar la venda) i informar a la tenda sobre la data d'avui.

Finalment establim els mètodes adients per ordenar items i vendes des de la classe principal. Necessitarem importar la classe **Collections** definida per defecte a Java que conté un mètode (**sort( )**) capaç d'ordenar una llista específica amb objectes que implementin la la interfície Comparable abans esmentada. Per ordenar productes definim **sortItems( )**, per ordenar vendes **sortSales( )**. Aquests mètodes cridaràn a sort( ) de Collections per aplicar el procediment indicat per la funció compareTo( ) respectiva de cara a ordenar-los. Finalment indica per pantalla el resultat de la distribució d'objectes ja ordenada.

POSSIBLES SOLUCIONS ALTERNATIVES

CALENDAR CLASS COM ALTERNATIVA A DATE

Per tal de millorar el maneig de les dates haviem, principalment, pensat en implementar la classe Date ja definida a Java. Un cop ja teniem plantejada la implementació del canvi ens vam adonar que alguns mètodes importants de la classe que ens calia usar estaven obsolets. Llavors és per aquest fet que hem decidit acabar implementant la classe Calendar; que compleix les mateixes tasques i és actual.

CONCEPTES TEÒRICS

Esmentarem , a part dels ja vistos en el laboratori anterior, alguns aspectes de la teoria que hem observat estan relacionats amb els conceptes de programació orientada a objectes que hem aplicat com a part de la solució:

Interfícies: aquest laboratori ens ha permés palpar la aplicació de diverses interfícies. Hem pogut observar que resulten útils a l'hora d'agrupar comportaments similars quan no podem fer-ho per herència, donada la restricció en Java de l'herència múltiple. També tornem a observar de nou, gràcies a les interfícies, les tècniques d'override de mètodes i la implementació de mètodes abstractes.

Prototyping: La classe Collections amb el mètode sort( ) pot ordenar una llista d'instàncies de qualsevol classe que implementi aquesta interfície. Observem doncs com definim les ordenacions d'items i vendes amb una implementació basada en la interfície Comparable.

EL NOSTRE PROCEDIMENT I IMPLICACIÓ DE CADA MEMBRE

Per aquest laboratori hem seguit un procediment similar a l'anterior. La gestió del mateix comença el dia de la sessió online del laboratori, on discutim conjuntament el disseny general de la solució i proposem com es podria realitzar en forma de codi en Java. Un cop tenim les idees clares i entenem com implementar la majoria dels punts a realitzar procedim a, per separat, implementar les solucions apartat per apartat.

Més tard tornem a discutir conjuntament els nostres avenços per decidir què implementar i avançar cap a la solució final. Un cop tenim una versió del programa que satisfà les motivacions del projecte decidim passar a la fase de "perfeccionament" on ens repartim el treball per deixar un codi entenedor; amb coherència i ben cohesionat. Després, finalment, procedim a comentar el mateix i a redactar aquest document.

CONCLUSSIÓ

El resultat final del laboratori és per nosaltres satisfactori, observem que el programa compleix satisfactòriament amb les tasques proposades i es mostra robust. Mitjançant els resultats dels tests de la classe OnlineStore podem observar que les classes estan ben implementades (Figura 2) i permeten aplicar l'impost de l'IVA, manegar dates, vendes... Compleix perfectament l'objectiu de perfeccionar la tenda online feta pel laboratori anterior. Hem vist com el projecte anterior era perfectament ampliable; la solució ha permés construir actualitzacions majors a partir de la versió anterior.

```
Box with size {200.0, 300.0, 500.0} assigned to item Volvo
Administrator Joel Peterson is printing the current stock:
Volvo has current price 10000.0 with auction deadline on day: 6 of December
day: 3 of December

Toni García makes a bid for the item Volvo of 11000.0 euros.
day: 4 of December

Pau Cobacho makes a bid for the item Volvo of 11500.0 euros.
day: 5 of December

Arnau Adan makes a bid for the item Volvo of 13000.0 euros.
day: 6 of December

Auction of Volvo is frozen. No more bids can be made.
Joel Peterson managed the item Volvo, Buyer: Arnau Adan
Arnau Adan is buying item Volvo for 13000.0 euros
Price 13000.0 is getting charged into account 453627 from user Arnau Adan
Item profit for the online store: 655.0
day: 7 of December

Toni García makes a bid for the item Volvo of 13500.0 euros.
Joel Peterson expelled the user Toni García
SORTED SALES LIST:
6 of December: Volvo bought by Arnau Adan
2 of December: Rice bought by Toni García
2 of December: TV bought by Arnau Adan
2 of December: Sofa bought by Pau Cobacho
STORE ANALYTICS:
Total items and packets price (+iva): 43862.5 euros.
Total items and packets profit: 2881.494666417931 euros.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figura 2: Captura de pantalla parcial sobre l'output de la solució.