Pau Cobacho, Arnau Adan
25 novembre de 2020
POO

Lab session 3: Implementing inheritance

OnlineStore

INTRODUCCIÓ DEL PROBLEMA

L'objectiu d'aquesta sessió de laboratori és implementar el disseny del seminari 3. La solució resultarà en una aplicació orientada a objectes que usarà el concepte d'herència. L'aplicació en questió és tracta d'una tenda online que vendrà una varietat de productes, nous i usats (de segona mà). Ens haurem de basar en el disseny del seminari per implementar les classes representant els usuaris connectant-se a la tenda i els productes sent venuts.

El disseny de classes proposat (Figura 1) es basa en quatre classes principals: User, Package, Item i OnlineStore. Les tres primeres tindràn classes filles que permetràn especialitzar-les, introduïnt uns comportaments i característiques diferents a cada derivat:

- User: representa l'usuari de la OnlineStore.
    - Buyer: usuari comprador.
    - Seller: usuari venedor.
    - Administrator: usuari administrador.
- Package: representa el paquet d'enviament de la OnlineStore.
    - Box: paquet del tipus caixa.
    - Envelope: paquet del tipus sobre.
- Item: representa el producte de la OnlineStore.
    - UnitItem: producte per unitat.
    - WeightedItem: producte per pes.
    - AuctionItem: producte de subhasta.
- OnlineStore: representa la infromació de la mateixa tenda.

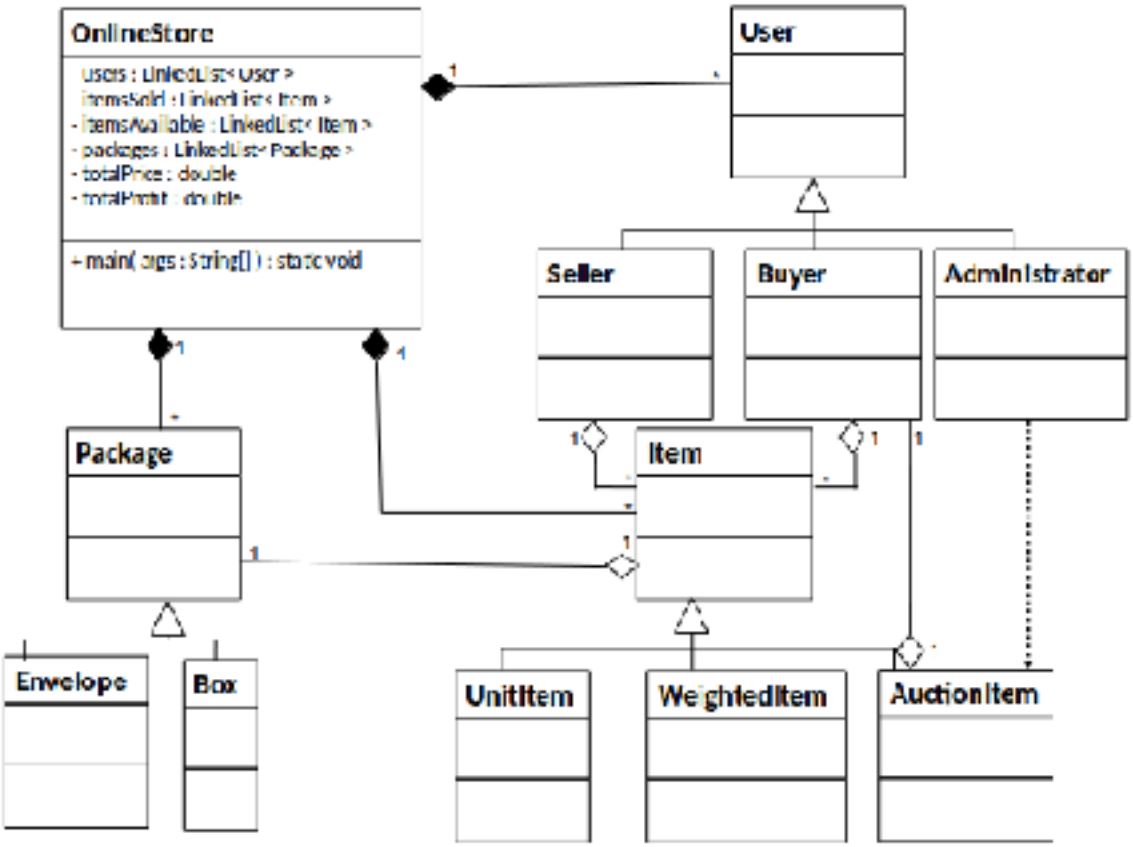

Figura 1: Diagrama de classes per la tenda

Per tant, necessitarem implementar, en total, 12 classes diferents. Haurem de manegar els diferents tipus d'usuaris, items i paquets per tal que la tenda funcioni perfectament i sugui capaç de, per exemple; vendre productes, seleccionar els paquets adients per aquests, realitzar subhastes... A continuació explicitem els detalls per cada classe, repassant els mètodes i atributs pertinents.

CLASSE ITEM

Cada Item serà representat per un nom, tipus (per exemple; un llibre, un juguet, un moble, etc.), unes dimensions aproximades i el seu paquet d'enviament adient. Tindrem tres categories diferents d'items, classificades segons com el seu preu i benefici són calculats (UnitItem, WeightedItem i AuctionItem).

Llavors els atributs d'item seràn 5: **name** (del tipus string representant el nom del producte) , **type** (del tipus string brindant el tipus), **size** (del tipus double[3] informant sobre el component en cm de cada dimensió del producte), **cost** (del tipus double) i **pack** (repressentant un objecte de la classe Package).

En quant als seus mètodes, per definir la morfologia dels items implementarem un constructor que assignarà els atributs a cada instància i també un que no n'assigni cap (tal és com es demana a l'enunciat). També establirem getters i setters que ens serviràn per obtenir i establir la informació sobre els atributs: **getName( )**, **getType( )**, **getSize( )**, **getCost( )**, **getPackage( )**, **setName( )**, **setType( )**, **setSize( )** i **setCost( )**.

Implementarem, a més a més, el mètode **assignBestPackage( )** capaç de seleccionar el paquet (de la classe Package) adient d'entre tots els disponibles segons el tipus d'item sobre el qual s'estigui treballant. També **getPrice( )** i **calculateProfit( )** que seràn mètodes abstractes capaços de calcular el preu i benefici per cada producte, els considerem d'aquest tipus ja que cada tipus d'item representat a les classes filles els implementarà de manera diferent, és a dir, calcularàn el cost i el preu diferentment.

Ara, detallem les classes filles representant cada tipus de producte:

- UNITITEM: l'objecte representa els productes venuts per unitat. Vindrà definit per les caràcteristiques del seu pare i pel seu preu per unitat **unitPrice**, quantitat en unitats **quantity** i quantitat restant **quantityRemaining**. Pel seu comportament definirem el constructor i les funcions abstractes mencionades anteriorment (fent així un "override") **getPrice( )** i **calculateProfit( )**. El preu es calcularà depenent de les quantitats desitjades per l'usuari mentres que el benefici resultarà la diferència entre el preu (més baix) pel qual la tenda ha obtingut el producte i el preu que l'usuari ha pagat per aquest (més alt). Afegim també **sell(** int q **)** que s'encarregarà d'actualitzar la quantitat actual del producte i retornarà el preu a pagar per totes les unitats existents.

- WEIGHTEDITEM: l'objecte representa els productes venuts per pes. Vindrà definit per les caràcteristiques del seu pare (nom, tipus, cost, mida i paquet assignat) i pel seu preu per pes **pricePerWeight**, quantitat total en pes **weight** i quantitat restant en pes **weightRemaining**. Pel seu comportament definirem el constructor i les funcions abstractes **getPrice( )** i **calculateProfit( )**. El preu es calcularà depenent del pes desitjat per l'usuari mentres que el benefici resultarà la diferència entre el preu/pes (més baix) pel qual la tenda ha obtingut el producte i el preu/pes que l'usuari ha pagat per aquest (més alt).

- AUCITONITEM: l'objecte representa els productes venuts per subhasta. Vindrà definit per les caràcteristiques del seu pare i pel seu preu actual en la subhasta **currentPrice**, el seu licitador acutal **bidder** (representat per la classe Buyer, filla d'User), la data límit de la subhasta **deadline** i una quota **fee** i un càrrec **charge** de 5 euros i 5% respectivament que haurà de respectar cada usuari al fer una subhasta. Per tant aquests atributs seràn de tipus final (sempre tindràn el mateix valor inamovible per cada objecte instanciat). Pel seu comportament definirem el constructor i les funcions abstractes **getPrice( )** i **calculateProfit( )**. El preu es calcularà en funció de la puja més elevada actualment mentres que el benefici resultarà dels 5 euros imposats pel càrrec i la quota del 5% del producte subhastat. Inclourem també comportaments per a que un comprador interessat pugui pujar per la instància d'objecte de subhasta actual **makeBid(** Buyer b, double p**)**, també per aturar una subhasta si la data límit ha arribat **frozen(** string d **)** i finalment getters per obtenir informació de la data límit i el licitador líder actualment de la subhasta **getBuyer( )** i **getDeadline( )**.

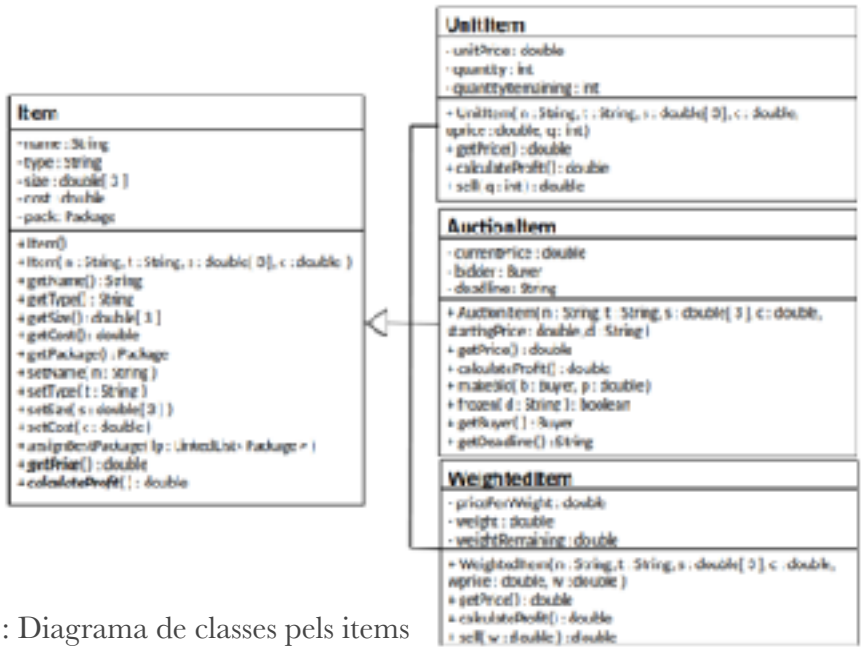

Figura 2: Diagrama de classes pels items

CLASSE USER

Cada usuari serà representat per un nom, identificador (usat per connectar-se) i contrassenya . Tindrem tres tipus d'usuaris inteactuant amb la tenda (compradors, venedors i administradors). Llavors els atributs d'User seràn tres: **name**, **identifier** i **password** (tots del tipus string).

En quant als seus mètodes, per definir la morfologia dels items implementarem un constructor que assignarà els atributs a cada instància. També establirem getters i setters que ens serviràn per obtenir i establir la informació sobre els atributs: **getName( )**, **getID( )**, **getPassowrd( )** i **setName( )**. Implementarem, a més a més, el mètode **login(** String p **)** que servirà per iniciar la sessió de la instància actual d'usuari si té la contrassenya correcta a l'argument de la funció.

Ara, detallem les classes filles representant cada tipus d'usuari:

- BUYER: l'objecte representa els usuaris compradors. Vindrà definit per les caràcteristiques del seu pare (nom, identificador i contrassenya) i pel seu número de compte bancari **accountNumber** i una llista amb els productes comprats **boughtIems**. Pel seu comportament definirem el constructor i les funcions **buy(** Item i **)** i **pay(** double price **)**. La primera permet a la instància d'usuari comprador actual comprar el producte (de la classe Item) indicat per l'argument, afegint-lo a la llista de productes comprats i descomptant-ne el seu preu del seu compte bancari. La segona s'encarrega d'indicar per pantalla l'acció de pagar per part del comprador.

- ADMINISTRATOR: l'objecte representa els usuaris administradors. Vindrà definit per les caràcteristiques del seu pare. Pel seu comportament definirem el constructor i les funcions **expel(** User u **)** que expulsa l'usuari indicat per l'argument de la funció en cas de mal comportament eliminant-lo de la llista d'usuaris de la tenda. **manageAuction(** AuctionItem a, String date **)** que permet a la instància actual d'administrador gestionar la subhasta indicada pel primer paràmetre i aturar-la si la data del segon paràmetre equival a la data límit de l'objecte de subhasta indicat. Finalment

afegim **printStock( )** que s'encarregarà d'informar sobre tots els elements actuals posats a subhastar.

- SELLER: l'objecte representa els usuaris venedors. Vindrà definit per les caràcteristiques del seu pare i pel seu número de compte bancari **accountNumber** i una llista amb els productes venuts i disponimbles **soldtIems** i **availableItems** respectivament. Pel seu comportament definirem el constructor i les funcions **sell(** Item i), **addAvailableItem(** Item i), **deposit(** double price). La primera s'encarregarà d'informar sobre l'acció de venta per part de la instància actual de venedor del producte indicat pel paràmetre, la segona afegirà a l'usuari venedor que cridi la funció l'item del paràmetre a la seva llista de productes disponibles. La tercera determinarà si l'usuari venedor desitja dipositar els diners obtinguts de la venta directament al compte bancari.

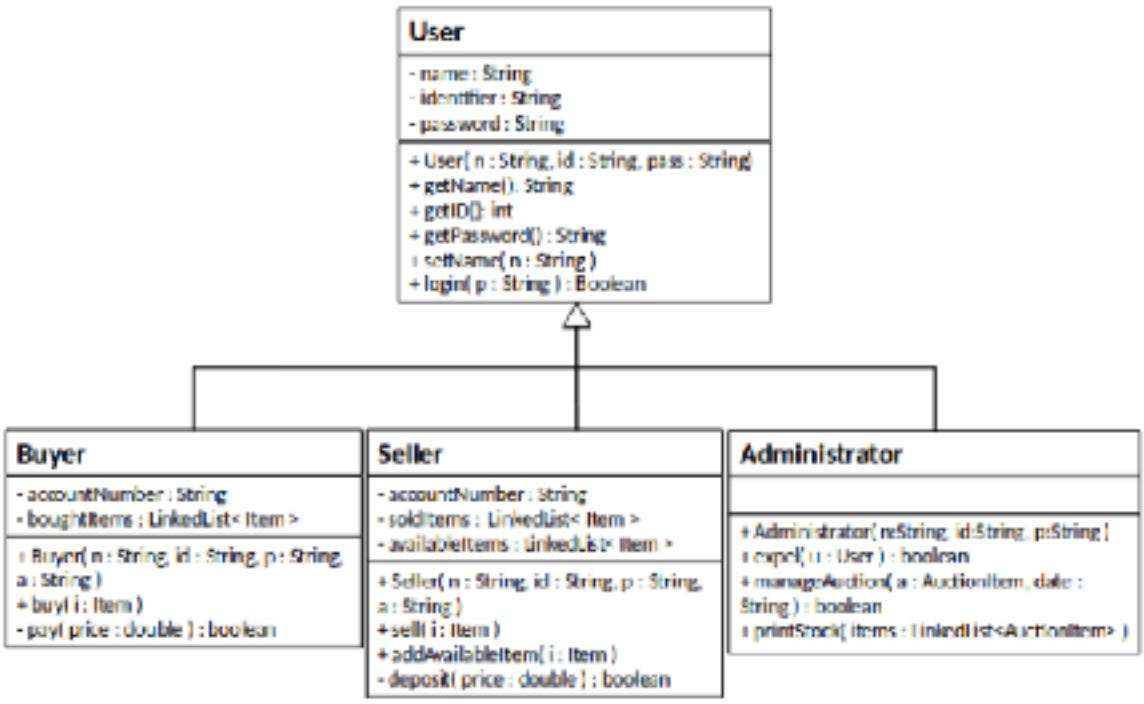

Figura 3: Diagrama de classes pels usuaris

CLASSE PACKAGE

Els items s'envien en paquets, cada paquet serà representat per les seves dimensions amplada i llargada i tindrem dos categories diferents d'aquests: Envelope i Box. O sigui, sobres i caixes. Llavors els atributs de Package seràn dos: **width**, **height**.

En quant als seus mètodes, per definir la morfologia dels paquets implementarem un constructor que assignarà els atributs a cada instància. També establirem getters i setters que ens serviràn per obtenir i establir la informació sobre els atributs: **getWidth( )**, **getHeight( )**, **setWidth( )** i **setHeight( )**.

Ara, detallem les classes filles representant cada tipus de paquet:

- ENVELOPE: l'objecte representa els paquets del tipus sobre. Vindrà definit per les caràcteristiques del seu pare (amplada i llargada) i pel seu nom **name**. Pel seu comportament definirem el constructor i les funcions **getName( )**, **setName( )** i **isSuitable(** int size[2] **)** que estudia si la instància actual de l'objecte sobre pot ser usada per enviar el producte amb les mides indicades pel paràmetre.

- BOX: l'objecte representa els paquets del tipus caixa. Vindrà definit per les caràcteristiques del seu pare i per la seva profunditat **depth**. Pel seu comportament definirem el constructor i les funcions **getDepth( )**, **setDepth( )** i **isSuitable(** int size[3] **)** que, de manera similar a l'anterior, estudia si la instància actual de l'objecte caixa pot ser usada per enviar el producte amb les mides indicades pel paràmetre.

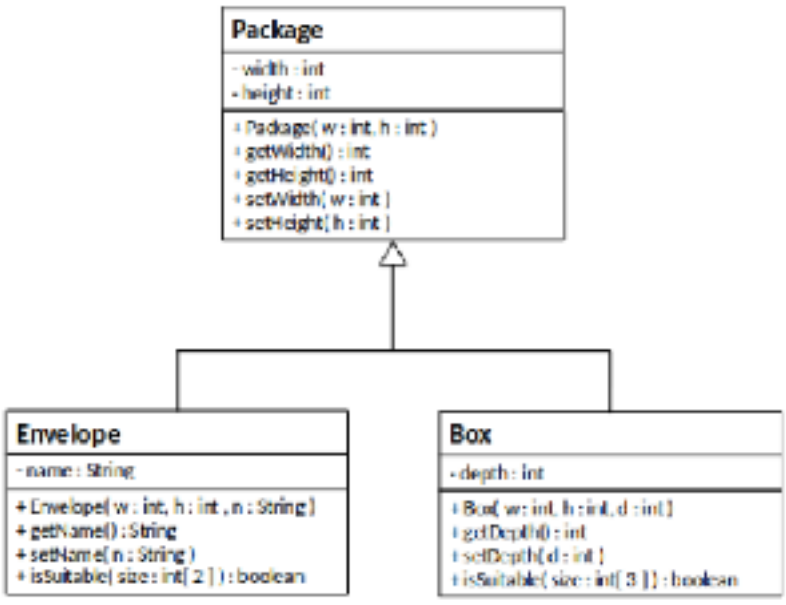

Figura 4: Diagrama de classes pels paquets

CLASSE ONLINESTORE

Serà l'objecte encarregat de manegar i dirigir la resta. Aqui podrem realitzar compres, ventes, subhastes, aplicar paquets als items, aturar subhastes, expulsar a usuaris... Per tant l'objecte que representa la tenda estarà format per llistes enllaçades representant els seus productes venuts i disponibles **itemsSold** i **availableItems**, més llistes definint els paquets disponibles i usuaris **packages** i **users**. També emmagatzemarà en tot moment el preu i benefici actuals **totalPrice** i **totalProfit**, que inicialment seràn els dos equivalents a 0. Cada cop que un producte sigui venut l'eliminarem de la llista de productes en venta i actualitzarem les variables que informen del preu i benefici de la tenda per mostrat els estalvis de la mateixa. Aquests, doncs, són els seus atributs. Pel seu comportament afegim un constructor **init( )** per inicialitzar els atributs de la tenda.

POSSIBLES SOLUCIONS ALTERNATIVES

No hem tingut gaires discussions sobre possibles solucions alternatives. Tot i així ens ha costat acabar definint el mètode **isSuitable( )** per la classe Box ja que no trobàvem cap implementació funcional. Finalment hem conseguit redactar un codi capaç de complir amb la tasca desitjada.

Per **assignBestPackage( )** hem decidit finalment optar per calcular l'àrea o volum de l'objecte per assignar-li el paquet correcte. S'havia optat per diferents procediments però finalment hem cregut aquets com el més elegant.

CONCEPTES TEÒRICS

Esmentarem 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:

Herència: Hem palpat clarament les implicacions i els beneficis d'aquest recurs. Observem com els "instance members" de les classes pare són aplicables perfectament a les seves filles i com promouen una solució elegant pel problema. Observem com permet especialitzar una superclasse.

Relacions entre classes: Observem les relacions entre els objectes de l'aplicació gràcies a les múltiples relacions que apareixen en el laboratori; per exemple, observant que la classe Item té com a atribut una variable del tipus objecte Package, mostrant una clara relació d'agregació. També, per exemple, que tant la classe Buyer com Seller emmagatzemen objectes del tipus Item en la seva llista d'atributs, mostrant de nou una relació d'agregació. També observem que la classe OnlineStore emmagatzema en tot moment els usuaris, items i paquets de la tenda, per tant, de nou veiem una relació entre les les classes d'aquests objectes (User, Item i Package) amb l'objecte OnlineStore, mostrant la relació de composició.

Classes i mètodes abstractes, override: Tenim la classe Item com a classe abtracta, això es deu a que els seus objectes fills empleen getPrice( ) i calculateBenefit( ) de diferents maneres, per tant, no cal definir el mètode a la funció pare. Tenim llavors un mètode sense implementar, per tant la classe es converteix en asbtracta al haver-hi en ella dos mètodes abstractes. Més tard, a les classes filles implementem auqests mètodes, fent un "override" (on conservem la signatura del mètode però modifiquem la implementació).

Upcast i downcast: Observem l'ús al voler modificar el tipus d'instància de la subclasse al de la super classe i viceversa.

instanceof: Observem el seu ús al, per exemple, comprovar a la classe OnlineStore si l'element comprat és un del caire unitat o pes i, un cop ho sabem, apliquem la operació de la classe corresponent a l'objecte.

keyword super: hem necessitat usar-la per cridar els atributs d'una superclasse a les seves classes filles.

EL NOSTRE PROCEDIMENT I IMPLICACIÓ DE CADA MEMBRE

Per tal de concretar la implicació de cada membre pel laboratori, a continuació expliquem el procés de gestació del mateix:

La gestació comença el dia de la sessió online del laboratori, on discutim conjuntament les parts menys clares sobre el disseny de la solució. Tot seguït decidim començar a programar la estructura principal de la solució; escrivint i implementant les funcions constructors, getters i setters, els atributs per cada objecte i les signatures de la resta dels mètodes. Un cop arribats aqui hem comparat els resultats obtinguts fins llavors de cadascun, després hem decidit anar "testejant" que les implementacions resultaven correctes. Tot seguit ens hem repartit les funcions restants per, al cap d'uns dies, mostrar les nostres aproximacions a les solucions de cadascuna i aplicar-les a la solució.

Un cop tenim el projecte avançat i més complert decidim començar a orquestrar el moviment de la tenda des de la classe OnlineStore. On, posant-nos d'acord acabem definint la versió final que dirigeix amb èxit els moviments de la tenda.

Un cop el codi s'ha mostrat funcional hem decidit comentar-lo per classes, assignant-ne cada classe a un membre específic. Després hem procedit, finalment, a la redacció d'aquest informe. Redactant un la introducció i l'altre la explicació per cada classe.

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 5). Compleix perfectament l'objectiu de modelar parcialment una tenda online.

El projecte ens ha permès comprovar de manera pràctica les virtuds de la programació orientada a objectes veient com els nostres objectes promouen els seus principis bàsics de la herència (promovent "reuse" al reusar la definició de la superclasse, evitant la ducplicació de mètodes) i el polimorfisme (mostrant com un mateix objecte pot comportar-se de maneres diverses depenent del context). Pensem que la solució permet construir actualitzacions majors a partir de la versió actual i pot ser usada per complir tasques diferents si és desitjat.

```
compile:
run:
Pau user account login was successfull
Toni user account login was successfull
Arnau user account login was successfull
Jose user account login was successfull
Jefe user account login failed, please try again
Box with size {100.0, 150.0, 300.0} assigned to item Sofa
Envelope A4 assigned to item Rice
Box with size {10.0, 180.0, 100.0} assigned to item TV
Pau Cobacho is buying item Sofa for 1000.0 euros
Price 1000.0 is getting charged into account 12345678 from user Pau Cobacho
All items left of Sofa have been sold
Jose Rodrigo sold Sofa and 400.0 euros are deposited to account 1111
Toni Garcia is buying item TV for 4000.0 euros
Price 4000.0 is getting charged into account 87654321 from user Toni Garcia
All items left of TV have been sold
Jose Rodrigo sold TV and 1600.0 euros are deposited to account 1111
Box with size {200.0, 300.0, 500.0} assigned to item Volvo
Toni Garcia is the actual highest bidder of the item Volvo
Administrator Joel Peterson is printing the current stock:
Volvo has current price 11000.0 with auction deadline 20201010
Arnau Adan is the actual highest bidder of the item Volvo
Joel Peterson managed the item Volvo, Buyer: Arnau Adan
Joel Peterson expelled the user Toni Garcia
Arnau Adan is buying item Volvo for 13000.0 euros
Price 13000.0 is getting charged into account 453527 from user Arnau Adan
Jose Rodrigo sold Volvo and 655.0 euros are deposited to account 1111
Total price: 18000.0
Total profit: 2655.0
BUILD SUCCESSFUL (total time: 3 seconds)
```

Figura 5: Mostra de l'output de la solució del laboratori.