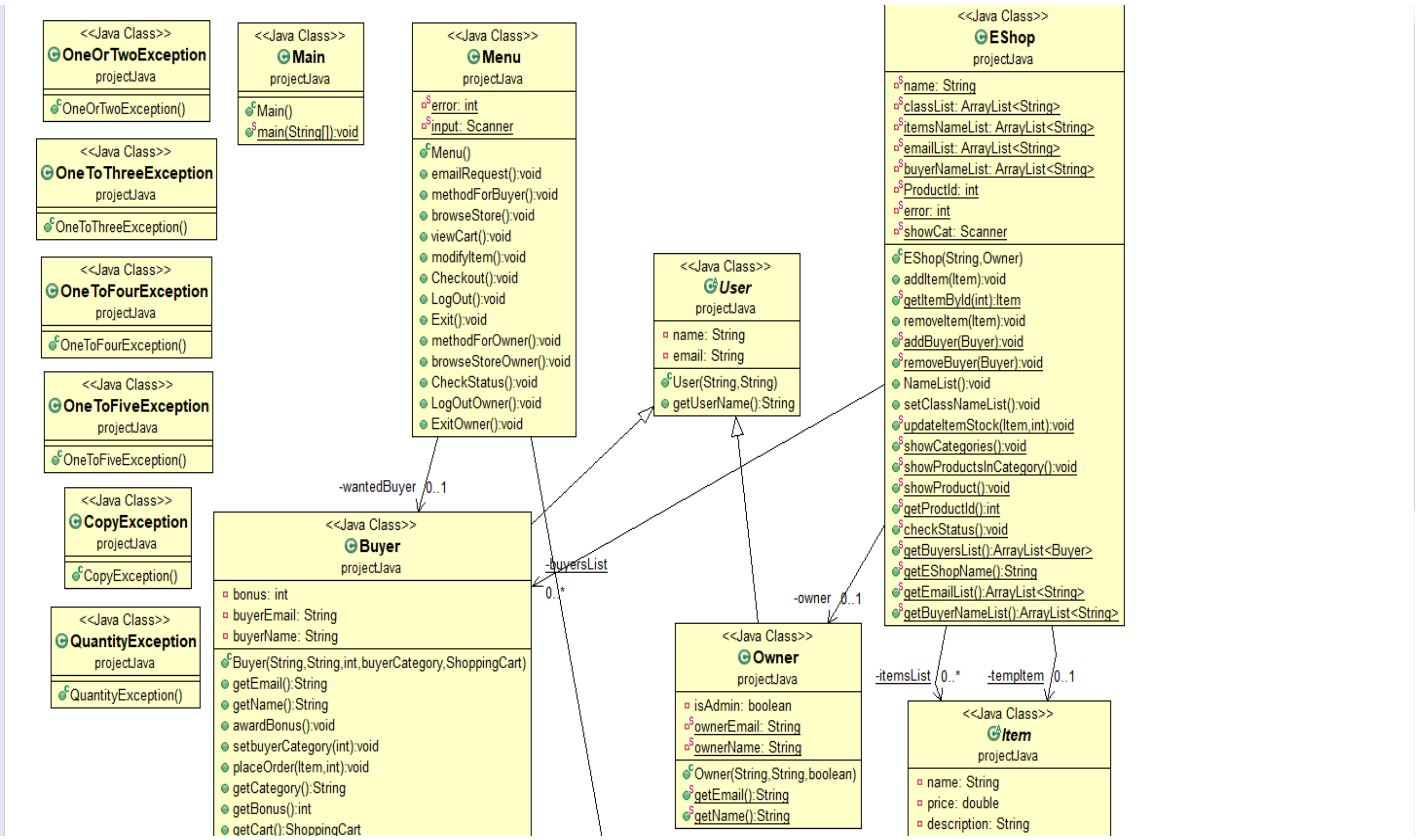
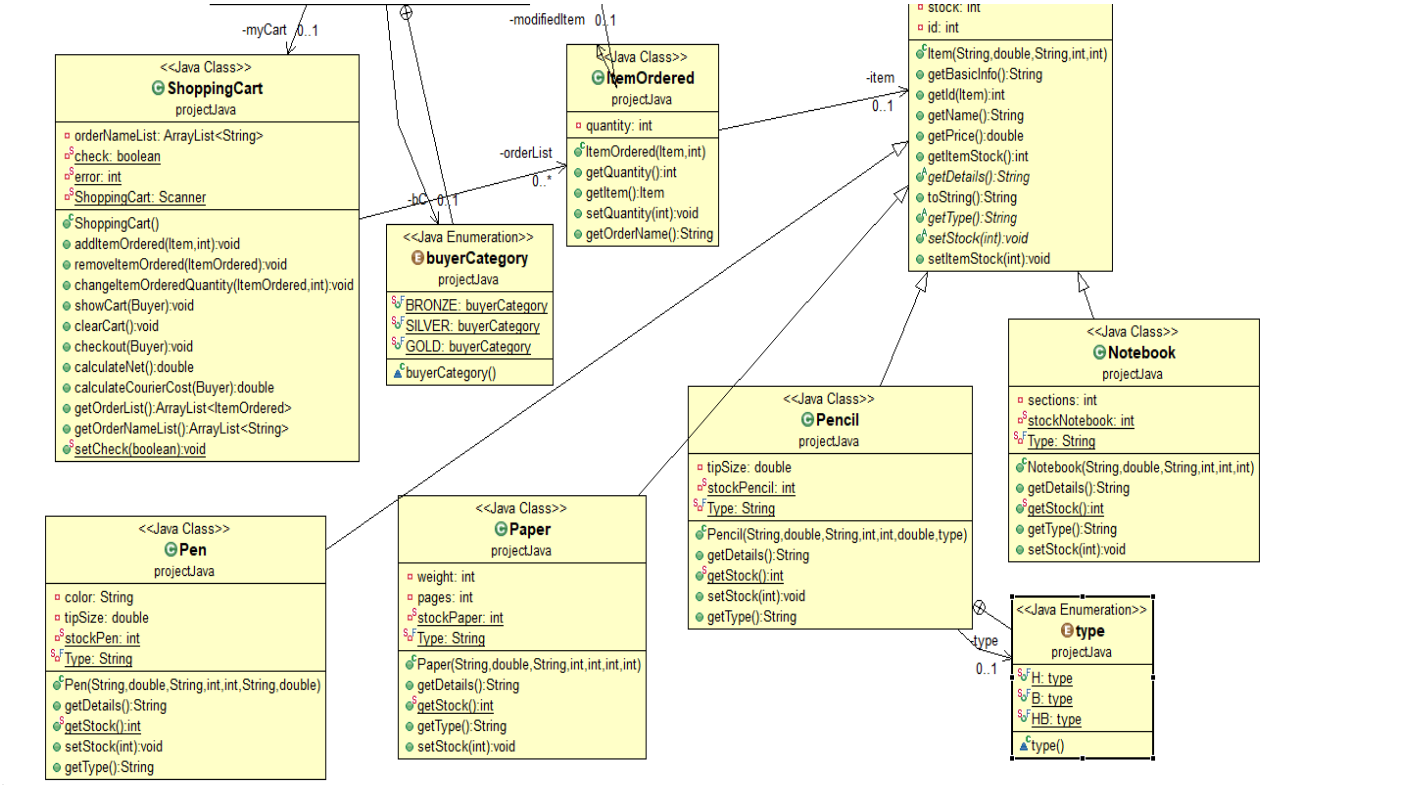
ProjectOO 2ου εξαμήνου 2019-2020 στον Οντοκεντρικό Προγραμματισμό

Δούρου Βασιλική Ευαγγελία - 1072633 - valiadourou@gmail.com
Μπαλάφας Δημήτριος - 1072499 - jimbalafas2001@gmail.com
Πεσκελίδης Παύλος - 1072483 – paulpesk@hotmail.gr

Διάγραμμα UML :





Περιγραφή των κλάσεων :

Κλάση User :

Η κλάση είναι abstract μιάς και δεν επιτρέπει στιγμιότυπα. Περιέχει ένα constructor με 2 μεταβλητές που χρησιμοποιείται από τις κλάσεις Buyer και Owner. Τέλος , έχει μία μέθοδο getUserName() ,η οποία επιστρέφει το string name.

Κλάση Owner :

Η Owner είναι υποκλάση της User και κληρονομεί τις μεταβλητές name, email. Παράλληλα μέσα στον constructor περιέχονται 2 νέες μεταβλητές string ownerEmail, ownerName , που λαμβάνουν τις τιμές των email και name αντίστοιχα. Ακόμη, δημιουργήθηκαν 2 νέες getters που επιστρέφουν τα ownerEmail, ownerName αντίστοιχα.

Κλάση Buyer :
Η User είναι υποκλάση της User και κληρονομεί τις μεταβλητές name, email. Περιέχει τις κατηγορίες BRONZE, SILVER, GOLD σε μορφή enum και πρόσθετες μεταβλητές τις εξής :
Μεταβλητή bc τύπου buyerCategory (enum).
Μεταβλητή myCart τύπου ShoppingCart .
Μεταβλητές buyerEmail, buyerName τύπου string.

Επίσης δημιουργήθηκαν οι μέθοδοι getName, getEmail (επιστρέφουν buyerName,buyerEmail) , awardBonus, setBuyerCategory (ανάλογα με τη τιμή του ορίσματος bonus αντιστοιχείται και μία buyerCategory). Τέλος, υπάρχουν, η getCategory που μετατρέπει τη bC σε string και οι getBonus , getCart (τυπου ShoppingCart) που επιστρέφουν τις bonus και myCart.

Κλάση Item :
Η Item περιέχει τις πρόσθετες μεθόδους getId με όρισμα Item I και επιστρέφει i.id, getName (επιστρέφει name), getPrice (επιστρέφει price), getItemStock (επιστρέφει stock). Παράλληλα υπάρχουν οι getDetails, getType, setStock(int s) χωρίς σώμα , οι οποίες χρησιμοποιούνται σε άλλες κλάσεις και η setItemStock με όρισμα int s που ικανοποιεί τη σχέση stock = stock + s.

Κλάση Pen :
Η Pen είναι υποκλάση της Item στην οποία δημιουργήσαμε 2 ακόμη μεταβλητές , την stockPen = 0 που αναπαριστά το συνολικό stock του αντικειμένου Pen και την (final String) Type = “Pen” ,η οποία χρησιμοποιείται για να έχουμε το όνομα του προϊόντος μέσα στο e-shop και καθώς έχει το final , το Type είναι σταθερό. Παράλληλα στον constructor εκτός από τις αναφερόμενες μεταβλητές υπάρχει και η stockPen που ικανοποιεί την σχέση stockPen=stockPen+stock. Δημιουργήθηκε η getStock (επιστρέφει το stockPen). Τέλος, γίνονται @Override οι setStock(int s) με σώμα stockPen=stockPen+s, όπου s μία νέα ποσότητα προϊόντος που προστίθεται στο stockPen και η getType ,η οποία επιστρέφει το Type.

Κλάση Pencil :
Η Pencil είναι υποκλάση της Item, στην οποία δημιουργήσαμε ένα enum type που περιέχει τους τύπους των μολυβιών (H,B,HB) και 2 ακόμη μεταβλητές την stockPencil = 0 που αναπαριστά το συνολικό stock του προϊόντος Pencil και την (final String) Type = “Pencil” ,η οποία μαζί με το final είναι μια σταθερά που χρησιμοποιείται για να έχουμε το όνομα του προϊόντος μέσα στο e-shop. Παράλληλα στον constructor εκτός από τις αναφερόμενες μεταβλητές υπάρχει και η stockPencil που ικανοποιεί την σχέση stockPencil=stockPencil+stock.Επίσης χρησιμοποιήθηκε η getStock που με πολυμορφισμό, επιστρέφει το stockPencil. Καταλήγοντας, γίνονται @Override η setStock(int s) με σώμα stockPen=stockPen+s, όπου s μία νέα ποσότητα προϊόντος που προστίθεται στο stockPen και η getType, η οποία επιστρέφει το Type.

Κλάση Notebook :
Η Notebook είναι υποκλάση της Item. Στο σώμα της δημιουργήσαμε 2 πρόσθετες μεταβλητές την 2 μεταβλητές την stockNotebook = 0 που αναπαριστά το συνολικό stock του προϊόντος Notebook και την (final String) Type = “Notebook” ,η οποία μαζί με το final είναι μια σταθερά που χρησιμοποιείται για να έχουμε το όνομα του προϊόντος μέσα στο e-shop. Ακόμη, στον δημιουργό εκτός από τις αναγκαίες μεταβλητές υπάρχει και η stockNotebook που ικανοποιεί την σχέση stockNotebook=stockNotebook+stock. Επιπρόσθετα, χρησιμοποιείται η getStock που επιστρέφει stockNotebook. Τέλος, γίνονται @Override η setStock(int s) με σώμα stockNotebook=stockNotebook+s, όπου s μία νέα ποσότητα προϊόντος που προστίθεται στο stockNotebook και η getType, η οποία επιστρέφει το Type.

Κλάση Paper :
Η Paper είναι υποκλάση της Item. Στο σώμα της δημιουργήσαμε τις μεταβλητές (final String) Type = “Paper” ,η οποία μαζί με το final είναι μια σταθερά που χρησιμοποιείται για να έχουμε το όνομα του προϊόντος μέσα στο e-shop και stockPaper = 0 που αναπαριστά το συνολικό stock του προϊόντος Paper. Επίσης, στον δημιουργό εκτός από τις αναγκαίες μεταβλητές υπάρχει και η stockPaper που ικανοποιεί την σχέση stockPaper=stockPaper+stock. Επιπρόσθετα,

χρησιμοποιείται η getStock που επιστρέφει stockPaper.Καταλήγοντας, οι παρακάτω μέθοδοι γίνονται @Override :

- η setStock(int s) με σώμα stockPaper=stockPaper+s, όπου s μία νέα ποσότητα προϊόντος που προστίθεται στο stockPaper και
- η getType, η οποία επιστρέφει το Type.

Κλάση ItemOrdered :

Η ItemOrdered περιέχει 4 πρόσθετες μεθόδους εκτός των μεταβλητών item και quantity , οι οποίες συμπεριλαμβάνονται στον δημιουργό. Η getQuantity που επιστρέφει το quantity , η getItem που επιστρέφει το item, setQuantity(int q),που προσθέτει μία ποσότητα προϊόντος στην quantity , ικανοποιώντας την σχέση quantity+=q και τέλος η getOrderName που επιστρέφει item.getName(), δηλαδή επιστρέφει το name του item που έχει παραγγείλει ο χρήστης από την κλάση Item.

Κλάση EShop :

Η EShop περιλαμβάνει τις μεταβλητές (String) name και (Owner) owner που βρίσκονται μέσα στον δημιουργό και την (Item) tempItem , η οποία χρησιμοποιείται μέσω της μεθόδου getItemById οταν διατρέχουμε την itemsList για να βρούμε αν υπάρχει το ζητούμενο αντικείμενο και αν υπάρχει αποθηκεύεται στο tempItem για να χρησιμοποιηθεί αργότερα. Παράλληλα, εκτός από τις ζητούμενες ArrayLists (itemsList,buyersList), δημιουργήθηκαν 4 νέες λίστες :

- classList : Περιέχει τα ονόματα των υποκλάσεων της κλάσης Item , δηλαδή των Pen, Pencil, Notebook, Paper. Σε περίπτωση που δεν έχει αντικείμενα κάποια από τις κλάσεις, τότε δεν εμφανίζεται η αντίστοιχη κατηγορία προϊόντος , ενώ αν το stock του είναι ίσο με 0 , εμφανίζεται διότι ο owner μπορεί να επιθυμεί να προσθέσει ο ίδιος stock.
- ItemsNameList: Περιέχει τα ονόματα των διαθέσιμων προϊόντων που υπάρχουν μέσα στις κατηγορίες του EShop.

- EmailList : Περιέχει μία λίστα των email των χρηστών που είναι εγγεγραμμένοι στο EShop και χρησιμοποιείται όταν εγγραφεται νέος χρήστης έτσι ώστε να καταχωρηθεί το νέο email.
- BuyerNameList : Περιέχει ένα σύνολο ονομάτων από τους χρήστες που είναι εγγεγραμμένοι και χρησιμοποιείται όταν ο owner επιθυμεί να δει πληροφορίες για τους χρήστες.

Τέλος, δημιουργήθηκαν οι μεταβλητές (int) ProductId για να μπορούμε να χρησιμοποιήσουμε το id, η (int) error που λειτουργεί ως activator για τις περιπτώσεις που έχουμε exceptions και ένα Scanner showCat που χρησιμοποιείται σε διάφορες μεθόδους.

Η μέθοδος addItem(Item i) περιέχει μία if , else έτσι ώστε καθώς διατρέχει την λίστα και βρει το αντικείμενο που προσπαθουμε να προσθέσουμε, στην λίστα να κάνει throw το CopyException (η χρήση του εξηγείται στη περιγραφή εξαιρέσαιων) , αλλιώς προσθέτει το αντικείμενο στην itemsList. Το σώμα της addItem περικλείεται σε μία do-while loop με την error=0 και να σπάει σε error=1 , έτσι ώστε να κάνει throw το Exception για όσο ο χρήστης δεν πράττει έγκυρα.

Η μέθοδος getItemById(int id), περιέχει μία for loop που ανατρέχει την itemsList για να βρει το ζητούμενο αντικείμενο , μέσω του id και να το αποθηκεύσει στο tempItem. Σε περίπτωση που δεν υπάρχει το id , η NoSuchElementException γίνεται throw. Τέλος, ισχύει το ίδιο με την addItem για το do-while loop με την error.

Η μέθοδος removeItem(Item i), λειτουργεί παρόμοια με την addItem , απλώς όταν διατρέχει την itemsList και αν δεν βρίσκει το item, κάνει throw NoSuchElementException ,ενώ αν το βρει καλεί την itemsList.remove(i).

Η μέθοδος addBuyer(Buyer b), λειτουργεί με τον ίδιο τρόπο με την addItem, ωστόσο αντί να διατρέχει την itemsList και να προσθέτει σε αυτήν, πραγματοποιεί αυτές τις λειτουργίες πάνω στην buyersList.

Όπως και η removeItem, η removeBuyer(Buyer b) διατρέχει την buyersList και αν δεν βρίσκει το Buyer b, κάνει throw NoSuchElementException ,ενώ αν το βρει καλεί την buyersList.remove(b).

Η NameList() διατρέχει την itemsList και για κάθε αντικείμενο της, παίρνει το name με την getName() και το προσθέτει στην itemsNameList.

Η setClassNameList() διατρέχει την itemsList και λαμβάνει τα ονόματα των κλάσεων , τα οποία τοποθετεί σε μία μεταβλητή (String) classes. Έπειτα, η classLists “σκανάρεται” και αν το όνομα που έχει αποθηκευτεί στη classes υπάρχει ήδη, κάνει throw το CopyException η οποία το διαγράφει από την λίστα, και προστίθεται το String ξανά.Σε περίπτωση που δεν υπάρχει, απλώς προστίθεται το όνομα.

Η updateItemStock(Item i,int stock) αποτελέιται από μια σειρά με 4 if-else if, όπου ελέγχεται σε ποιό stock αναφέρεται η ενέργεια μέσω του Type που έχει δωθεί σε κάθε κατηγορία. Όταν βρεθεί η κατάλληλη κατηγορία, καλείται η setStock στη σωστή κλάση.

Η showCategories() αρχικά ελέγχει αν η classList είναι κενή , όπου θα κάνει throw NoSuchElementException και θα τερματίσει το πρόγραμμα, καθώς το κατάστημα είναι άδειο. Επιπρόσθετα,μέσω ενός (String) s διατρέχει την classList και για όσες κλάσεις περιέχουν αντικείμενα , η μέθοδος εμφανίζει το ονόματα των κλάσεων και από δίπλα το stock που τους αντιστοιχεί.

Η showProductsInCategory() αρχικά ζητάει να δωθεί όνομα μίας κατηγορίας και η επιλογή αποθηκεύεται στην (String) choice. Έπειτα, γίνονται 5 έλεγχοι εκ των οποίων ο πρώτος ενεργοποιείται αν το choice δεν αντιστοιχεί στο όνομα κάποιας κατηγορίας ,όπου τότε γίνεται throw NoSuchElementException. Οι υπόλοιποι 4 έλεγχοι ακολουθούν την ίδια λογική : Σε περίπτωση που το choice είναι ίδια με το όνομα μίας από τις 4 κατηγορίες, τότε διατρέχεται η itemsList και για κάθε αντικείμενο που είναι στιγμιότυπο εκείνης της κατηγορίας εκτυπώνεται αριθμημένα το όνομά του.

Η showProduct() αρχικά ζητάει να δωθεί όνομα ενός προϊόντος και η επιλογή αποθηκεύεται στην (String) choice. Έπειτα , σκανάρει την itemsList και αν το προϊόν υπάρχει , το εμφανίζει και αποθηκεύει το id του σε μία μεταβλητή ProductId.Σε περίπτωση που το stock δεν είναι διαθέσιμο , εμφανίζει μήνυμα “This item is out of stock”.Τέλος, σε περίπτωση που δε βρεθεί το όνομα , γίνεται throw NoSuchElementException.

Η getProductId επιστρέφει ProductId.

Η checkStatus διατρέχει τη buyersList και εκτυπώνει αριθμημένους τους buyers με τα στοιχεία τους.

Η getBuyersList επιστρέφει buyersList

Η getEshopName επιστρέφει το όνομα του EShop.

Η getEmailList διατρέχει τη buyersList και προσθέτει στην emailList τα email των buyers και έπειτα επιστρέφει την EmailList.

Η getBuyerNameList διατρέχει τη buyersList και προσθέτει στην buyerNameList τα name των buyers και έπειτα επιστρέφει την buyerNameList.

Κλάση ShoppingCart :

Η ShoppingCart περιέχει μία πρόσθετη λίστα εκτός από την orderList, την (String) orderNameList , τις μεταβλητές (int) error=0 ,(boolean) check = false, το Scanner ShoppingCart που χρησιμοποιείται σε διάφορες μεθόδους, και το δημιουργό ShoppingCart ο οποίος είναι κενός.

Στη μέθοδο addItemOrdered(Item i ,int quantity) αρχικά δημιουργείται ένα αντικείμενο order τύπου ItemOrdered , το οποίο μετά προστίθεται στην orderList. Τέλος , καλούνται οι updateItemStock και setItemStock με όρισμα -quantity αλλάζοντας έτσι το stock.

Στη μέθοδο removeItemOrdered(ItemOrdered i) γίνεται αρχικά ένας έλεγχος της κατάστασης της μεταβλητής check. Αν η check = false καλείται η updateItemStock(i.getItem(),quantity), ενώ αν check=true καλεί την updateItemStock(i.getItem(),0) (η συγκεκριμένη κλήση χρησιμοποιείται όταν ο χρήστης θέλει να κάνει checkout και δεν θέλουμε να καλείται η updateItemStock πάλι). Τέλος καλείται η remove(i) στην orderList για να αφαιρέσουμε τα ζητούμενα προϊόντα και η remove(i.getOrderName()) στην orderNameList για να αφαιρεθούν και τα ονόματα των προϊόντων.

Η changeItemOrderedQuantity(ItemOrdered i,int quantity) διατρέχει την orderList και αν βρει το (ItemOrdered) i καλεί την setQuantity στο i με όρισμα το ίδιο quantity και καλεί την updateItemStock για το i.getItem και για το -quantity έτσι ώστε να αφαιρεθεί από το ολικό stock.

Η showCart(Buyer b) διατρέχει την orderList μέσω ενός αντικειμένου (ItemOrdered) i ,εντοπίζει όλα τα αντικείμενα μέσα στη λίστα και τα εκτυπώνει

μαζί με τη τιμή τους. Τέλος, εκτυπώνει το συνολικό κόστος μέσω κλήσης της calculateNet() και το κόστος των μεταφορικών καλώντας την calculateCourrierCost(b).

Η clearCart() περιέχει μία if που ελέγχει αν η orderList είναι κενή. Σε περίπτωση που περιέχει αντικείμενα, καλεί την removeItemOrdered(orderList.get(0)) στην orderList και διαγράφει το στοιχείο στη πρώτη θέση. Η παραπάνω κλήση βρίσκεται σε while loop και ισχύει όσο η λίστα περιέχει αντικείμενα. Ωστόσο, στην περίπτωση που η orderList είναι NULL , γίνεται throw NullPointerException.

Η μέθοδος checkout(Buyer b) αρχικά καλεί τη showCart για να δείξει στον χρήστη τι έχει τοποθετηθεί στο καλάθι του μέχρι στιγμής και ερωτάται εάν επιθυμεί να συνεχίσει με το checkout. Σε περίπτωση, που ο χρήστης πατήσει “1” καλείται η awardBonus , Η μεταβλητή check γίνεται true έτσι ώστε να χρησιμοποιηθεί στην removeItemOrdered() που έμμεσα καλείται από την clearCart(). Σε περίπτωση, που πατηθεί το “2”, καλείται και πάλι η showCart(b). Τέλος, η μεταβλητή check γίνεται false,ενώ εάν δεν πατηθούν το 1 ή 2 γίνονται throw η OneOrTwoException (η χρήση της εξηγείται παρακάτω) και η InputMismatchException.

Η calculateNet() περιέχει μια μεταβλητή (double) total = 0.0. Στο εσωτερικό της μεθόδου ανατρέχεται η orderList και για το κάθε αντικείμενο η τιμή του (μέσω της getPrice()) πολλαπλασιάζεται με τη ποσότητα (μέσω της getQuantity()) και το αποτέλεσμα αποθηκεύεται στην total. Τέλος, επιστρέφεται η μεταβλητή total.

Η μέθοδος calculateCourrierCost(Buyer b) στο σώμα της έχει 2 νέες μεταβλητές. Την (double) cost που ικανοποιεί τη σχέση cost=calculateNet()*0.02 και την (String) category που ικανοποιεί τη σχέση category=b.getCategory(). Ακολουθεί, ένα switch-case block πάνω στις κατηγορίες που λαμβάνουμε από την getCategory.

Για case “BRONZE” : εάν το cost είναι ανάμεσα στο 0 και 3 , τότε το cost γίνεται 3 , αν είναι μεγαλύτερο ή ίσο του 3 ή ίσο με μηδέν , το cost παραμένει ίδιο .
Για case “SILVER” : το cost λαμβάνει το ήμισυ της τιμής (πολλαπλασιάζεται επί 0.5)
Για case “GOLD” : το cost λαμβάνει τη τιμή 0.
Καταλήγοντας, επιστρέφεται το cost.

Η μέθοδος getOrderList() επιστρέφει την orderList.

Η getOrderNameList() διατρέχει την ItemOrdered και για κάθε αντικείμενο της καλεί την getOrderName προσθέτοντας κάθε όνομα παραγγελίας στην λίστα orderNameList.Τέλος, επιστρέφεται η orderNameList.

Η setCheck(boolean b) χρησιμοποιείται για να θέσουμε τη τιμή της μεταβλητής check.

Κλάση Menu :
Η κλάση περιέχει τις μεταβλητές (Buyer) wantedBuyer , (int) error = 0 που έχει την ίδια χρήση με αυτήν σε προηγούμενες κλάσεις, (ItemOrdered) modifiedItem και ένα Scanner input και τέλος έναν constructor με το μοναδικό περιεχόμενο , την κλήση της emailRequest().

Η μέθοδος emailRequest αρχικά ζητά από τον χρήστη ένα email. Μέσω της getEmailList(), διατρέχεται η emailList και αν βρεθεί αντιστοιχία με το email που δόθηκε, “σκανάρεται” η buyersList , έτσι ώστε να βρεθεί το UserName , το bonus και η κατηγορία του χρήστη με το αντίστοιχο email. Τα παραπάνω στοιχεία εμφανίζονται στο terminal με κατάλληλο μήνυμα και καλείται η methodForBuyer. Ωστόσο, σε περίπτωση που το email ανήκει στον owner, εμφανίζεται το όνομα του με κατάλληλο μήνυμα και την methodForOwner(). Τέλος, σε περίπτωση που το email δεν υπάρχει πουθενά γίνεται throw η NoSuchElementException. Στη συγκεκριμένη περίπτωση , η εξαίρεση είναι

τροποποιημένη για να ρωτήσει το χρήστη αν θέλει να δημιουργήσει ένα νέο email και να δώσει τα κατάλληλα στοιχεία έτσι ώστε να προστίθεται στη buyersList.

Η methodForBuyer καλεί την getEshopName() για να εμφανιστεί το όνομα του Eshop. Έπειτα, εμφανίζονται οι επιλογές που έχει ο χρήστης και ακολουθεί ένα switch-case block. Από τη στιγμή που οι επιλογές είναι 5 σε περίπτωση που δεν είναι αριθμός μεταξύ 1-5 το input, γίνεται throw OneToFiveException και σε περίπτωση που δεν είναι αριθμός γίνεται throw η InputMismatchException.

- Αν ο χρήστης πατήσει 1, καλείται η browseStore().
- Αν ο χρήστης πατήσει 2, καλείται η viewCart().
- Αν ο χρήστης πατήσει 3, καλείται η Checkout().
- Αν ο χρήστης πατήσει 4, καλείται η LogOut().
- Αν ο χρήστης πατήσει 5, καλείται η Exit().

Η browseStore() αρχικά καλεί τις showCategories(), showProductsInCategory(), showProduct(). Αμέσως μετά, τη showProduct η μεταβλητή chosenItem λαμβάνει τιμή (ένα προϊόν) μέσω της getItemById και αν το stock αυτού του προϊόντος είναι ≤ 0 , τότε καλείται ξανά η browseStore. Στην αντίθετη περίπτωση ο χρήστης ερωτάται αν θέλει να προσθέσει το item στο καλάθι του (1.ναι ή 2.όχι).
Για “1” : ζητείται η ποσότητα που επιθυμεί να αγοράσει. Εάν είναι μεγαλύτερη του stock (ή μικρότερη του < 0) γίνεται throw QuantityException, ενώ αν υπάρχει ήδη στο καλάθι γίνεται throw CopyException(Η CopyException στη περίπτωση αυτή, εκτυπώνει κατάλληλο μήνυμα ωστόσο ανανεώνει το stock και στο καλάθι πια υπάρχει η νέα ποσότητα). Σε κάθε άλλη περίπτωση καλείται η placeOrder(chosenItem,quantity). Τέλος, ο χρήστης ερωτάται αν θέλει να συνεχίσει να κοιτάει προϊόντα , αν ναι καλείται η browseStore(), αν όχι καλείται η viewCart().

Για “2” : το πρόγραμμα ρωτά αν ο χρήστης επιθυμεί να γυρίσει πίσω. Εάν η απάντηση είναι 1(ναι) καλείται η methodForBuyer() , αν είναι 2(όχι) καλείται η brosweStore. Ολόκληρο, το σώμα του case περικλείεται σε ένα try-catch block με την OneOrTwoException.

Η viewCart() πρώτα εμφανίζει το καλάθι του χρήστη και έπειτα τον ρωτάει αν θέλει
1. να τροποποιήσει ένα προϊόν
2. να καθαρίσει το καλάθι του
3. να πάει στο checkout
4. να πάει πίσω
Σε περίπτωση που πατηθεί κάτι εκτός των 1,2,3,4 γίνεται throw η OneToFourException.
Για “1” : καλείται η modifyItem().
Για “2” : ερωτάται αν θέλει να καθαρίσει το καλάθι (1.ναι 2.όχι). Σε περίπτωση που πατηθεί κάτι εκτός των 1,2 γίνεται throw η OneOrTwoException.Αν πατηθεί το 1, καλείται η clearCart() και ανανεώνεται το stock του κάθε προϊόντος , ενώ αν πατηθεί το 2, καλείται η viewCart(). Τέλος, δίνεται η δυνατότητα του χρήστη αν θέλει να πάει πίσω (1.ναι 2.όχι) και πάλι αν δεν πατηθεί το 1 ή το 2 , γίνεται throw OneOrTwoException. Για 1 καλείται η methodForBuyer για 2 καλείται η viewCart.
Για “3” : καλείται η Checkout().
Για “4” : ο χρήστης ερωτάται αν θέλει να πάει πίσω (1.ναι 2.όχι). Αν δεν πατηθεί το 1 ή το 2 , γίνεται throw OneOrTwoException, αν πατηθεί τιμή λάθος τύπου γίνεται throw InputMismatchException. Για 1 καλείται η methodForBuyer για 2 καλείται η viewCart.

Στη modifyItem() πρώτα υπάρχει ένας έλεγχος εάν το καλάθι είναι κενό (δηλαδή η orderList), όπου θα τυπώσει κατάλληλο μήνυμα. Ύστερα, (αν η λίστα δεν είναι κενή) το πρόγραμμα ζητάει από τον χρήστη ποιό item επιθυμεί να τροποποιήσει και στη περίπτωση που το string που θα δωθεί αντιστοιχεί σε κάποιο item στην orderNameList , τότε τον ρωτάει εάν θέλει να 1. το διαγράψει , 2.αλλάξει την ποσότητα, 3.πάει πίσω.
Σε περίπτωση που η επιλογή του χρήστη δεν είναι 1,2,3 γίνονται throw OneToThreeException και InputMismatchException η οποία κάνει throw την NoSuchElementException.

Για “1” : καλεί την removeItemOrdered() για το item που ζητήθηκε και αμέσως μετά ρωτάει τον χρήστη αν θέλει να πάει πίσω (1.ναι 2.όχι), για 1 καλεί την methodForBuyer() για 2 την viewCart()
Για “2” : ρωτάει τον χρήστη αν θέλει να 1.αυξήσει τη ποσότητα 2.μειώσει τη ποσότητα ή 3.πάει πίσω. Και πάλι για input εκτός των 1,2,3 έχουμε τις κατάλληλες εξαιρέσεις. Το πρόγραμμα ζητάει από τον χρήστη τη ποσότητα που θα προσθέσει ή αφαιρέσει. Σε περίπτωση που η τιμή είναι μεγαλύτερη από το stock(και >0) ή η απόλυτη τιμή της μεταβλητής (όταν είναι <0) είναι μεγαλύτερη του stock, διεγείρεται η QuantityException. Για 1(αύξηση) καλείται η setItemStock() , η changeItemOrderedQuantity και η viewCart() , ομοίως για 2(μείωση).Για 3, καλείται η modifyItem(). Τέλος, υπάρχουν και πάλι εξαιρέσεις για τις 3 επιλογές
Για “3” : καλείται η modifyItem().

Στη μέθοδο Checkout() το πρόγραμμα εκτυπώνει το κόστος της παραγγελίας και των μεταφορικών μέσω των calculateNet() και calculateCourierCost() και καλεί την checkout() σε όρισμα wantedBuyer. Επίσης δίνεται η δυνατότητα να πάει πίσω ενώ παράλληλα υπάρχει OneOrTwoException(). Αν πατηθεί το 1 , καλείται η methodForBuyer, αν πατηθεί 2 , δημιουργείται το ερώτημα αν θέλει να 1.κάνει logout , όπου καλείται η LogOut() ή 2.να πάει στο μενού , όπου καλείται η methodForBuyer().

Αρχικά, στη LogOut() ζητάται επιβεβαίωση αν θέλει να κάνει log out (1.ναι 2.όχι) , η οποία συνοδεύεται από OneOrTwoException. Για 1 καλείται η emailRequest(), ενώ για 2 καλείται η methodForBuyer().

Στη μέθοδο Exit() αρχικά υπάρχει και πάλι μήνυμα επιβεβαίωσης με OneOrTwoException. Για 1 γίνεται System.exit(0) , ενώ για 2 καλείται η methodForBuyer.

Η methodForOwner διαφέρει μόνο στις επιλογές που δίνονται στο Menu(και στην αντίστοιχη εξαίρεση, η OneToFourException).Το menu του Owner είναι:

- Για 1, καλείται η browseStore().
- Για 2, καλείται η CheckStatus().
- Για 3, καλείται η LogOutOwner().
- Για 4, καλείται η ExitOwner().

Η browseStoreOwner διαφοροποιείται από την browseStore αμέσως μετά την επιλογή του προϊόντος, καθώς ρωτά τον owner αν επιθυμεί να τροποποιήσει το συγκεκριμένο προϊόν (1.ναι 2.όχι) .
Για "1" : εκτυπώνεται το αίτημα να δωθεί μία ποσότητα (αν >0) να προστεθεί ή (αν <0) να αφαιρεθεί. Αν η τιμή είναι <0 και μεγαλύτερη του stock κατ' απόλυτη τιμή, διεγείρεται QuantityException , αλλιώς καλούνται οι updateItemStock() και setItemStock(). Έπειτα ο ιδιοκτήτης ερωτάται αν θέλει να συνεχίσει να κοιτάει προϊόντα στο κατάστημα, για 1 καλείται η browseStoreOwner() για 2 η methodForOwner().
Για "2" : ζητείται επιβεβαίωση για το αν επιθυμεί να πάει πίσω. Για 1, καλείται η methodForOwner(), για 2 browseStoreOwner().

Η δεύτερη επιλογή του ιδιοκτήτη, είναι η CheckStatus(), η οποία αρχικά ρωτάει τον χρήστη αν θέλει 1.να τροποποιήσει κάποιον αγοραστή ή 2.να πάει πίσω.Αν επιλεγεί κάποια τιμή εκτός των 1 και 2 γίνεται throw OneOrTwoException.
Για "1" : γίνεται έλεγχος αν η buyersList είναι NULL , όπου θα ενημερώσει τον χρήστη και θα καλέσει την methodForOwner(). Αν δεν είναι κενή, ζητάει το όνομα του χρήστη προς τροποποίηση στο οποίο σημείο, διατρέχει τις buyerNameList και buyersList για να βρει τον αγοραστή. Σε περίπτωση που ο αγοραστής δεν υπάρχει , γίνεται throw NoSuchElementException.Έπειτα, καλείται η showCart() για τον ίδιο και τίθεται αίτημα για τον owner αν θα διαγράψει τον buyer (1.ναι 2.όχι με OneOrTwoException). Για 1 καλείται η removeBuyer() η setCheck(false) , η clearCart() και η CheckStatus() καθώς μπορεί να επιθμεί να τροποποιήσει παραπάνω από έναν buyer, ενώ για 2 καλείται η CheckStatus().

Για “2” : καλείται η methodForOwner().

Η LogOutOwner() είναι ίδια με την LogOut() ,αλλά για είσοδο “2”, καλείται η methodForOwner().

Η ExitOwner() είναι ίδια με την Exit() ,αλλά για είσοδο “2”, καλείται η methodForOwner().

Κλάση Main :
Στην κλάση main δημιουργούμε αντικείμενα τύπου Owner,Eshop και Pencil, Notebook, Pen, Paper. Ύστερα χρησιμοποιούμε την μέθοδο της EShop addItem() για να προσθέσουμε τα προϊόντα στο κατάστημα και χρησιμοποιούμε τις NameList() και SetClassNameList() για να προσθέσουμε αντικείμενα στην itemsNameList και classList. Μετά δημιουργούμε αντικείμενα ShoppingCart και Buyer και χρησιμοποιούμε την placeOrder() με τυχαία ορίσματα τύπου item και με τυχαίο quantity για να “γεμίσουμε” το καλάθι των Buyer που δημιουργήσαμε. Τέλος δημιουργόυμε ένα στιγμιότυπο του menu.

Κλάσεις Εξαιρέσεων :

Έχουν δημιουργηθεί επιπλέον 6 κλάσεις εξαιρέσεων ώστε να φαίνεται ευδιάκριτα ο λόγος για τον οποίο εγείρονται σε κάθε σημείο του κώδικα, ακόμη και αν δεν έχουν σώμα. Αυτές είναι: η CopyException, η οποία εγείρεται όταν υπάρχει ήδη ένα αντίγραφο σε κάποια λίστα, η QuantityException, η οποία εγείρεται όταν η δοθείσα ποσότητα δεν είναι διαθέσιμη (είτε επειδή είναι αρνητική είτε επειδή ξεπερνάει το στοκ) , η OneOrTwoException, η οποία εγείρεται όταν από επιλογή 1 ή 2 κάποιος τοποθετήσει άλλο αριθμό, η OneToThreeException, η οποία εγείρεται όταν κάποιος δεν επιλέξει 1,2 ή 3, η OneToFourException, που εγείρεται όταν ο χρήστης έχει να επιλέξει ανάμεσα σε

1,2,3 ή 4 και βάλει άλλον αριθμό και η OneToFiveException σεπερίπτωση που από τις επιλογές 1,2,3,4 και 5, ο χρήστης πληκτρολογήσει άλλον αριθμό.

Link Κώδικα : https://upatrasgr-my.sharepoint.com/:u:/g/personal/up1072633_upatras_gr/EXRE1t36a_dHhjoI4DL2eqIBhyAotyHHK6b4_5w6XeCuqA

Ο κώδικας είναι γραμμένος στο eclipse και είναι πλήρως λειτουργικός, σύμφωνα με τις απαιτήσεις σε επίπεδο λειτουργικότητας που φαίνονται στην εργασία.

Κώδικας ανά κλάση :

Κλάση User :

```
package projectJava;
public abstract class User {
	private String name;
	private String email;
	public User(String name,String email)
	{
		this.name=name;
		this.email=email;
	}
	public String getUserName()
	{
		return name;
	}
}
```

Κλάση Owner :

```
package projectJava;
public class Owner extends User{
	private boolean isAdmin=true;
	private static String ownerEmail,ownerName;
	public Owner(String name,String email,boolean isAdmin)
	{
		super(name,email);
		ownerEmail=email;
		ownerName=name;
```

```
		this.isAdmin=isAdmin;
	}
	public static String getEmail()
	{
		return ownerEmail;
	}
	public static String getName()
	{
		return ownerName;
	}
}
```

Κλάση Buyer :

```
package projectJava;
public class Buyer extends User {
	private int bonus=0;
	enum buyerCategory{
			BRONZE,
			SILVER,
			GOLD
	}
	private buyerCategory bC;
	private ShoppingCart myCart;
	private String buyerEmail,buyerName;
	public Buyer(String name,String email,int bonus,buyerCategory bC,ShoppingCart
myCart)
	{
		super(name,email);
		buyerEmail=email;
		buyerName=name;
		this.bonus=bonus;
		this.bC=bC;
		this.myCart=myCart;
	}
	public String getEmail()
	{
		return buyerEmail;
	}
	public String getName()
	{
		return buyerName;
	}
	public void awardBonus()
	{
		bonus+=(int)0.1*myCart.calculateNet();
	}
	public void setbuyerCategory(int bonus)
	{
		if (bonus<100)
		{
		bC = buyerCategory.BRONZE;
```

```
		}
		else if((bonus>100)&&(bonus<200))
		{
			bC = buyerCategory.SILVER;
		}
		else if(bonus>200)
		{
			bC = buyerCategory.GOLD;
		}
	}
	public void placeOrder(Item i,int quantity)
	{
		myCart.addItemOrdered(i,quantity);
	}
	public String getCategory()
	{
		return bC.toString();
	}
	public int getBonus()
	{
		return bonus;
	}
	public ShoppingCart getCart()
	{
		return myCart;
	}
}
```

Κλάση Item :

```
package projectJava;
public abstract class Item {
	private String name;
	private double price;
	private String description;
	private int stock;
	private int id;
	public Item(String name,double price,String description,int stock,int id)
	{
		this.name=name;
		this.price=price;
		this.description=description;
		this.stock=stock;
		this.id=id;
	}
	public String getBasicInfo()
	{
		return "Name: "+name+" Price: "+Double.toString(price)+" Description:
"+description+" Available stock: "+Integer.toString(stock)+" ID:
"+Integer.toString(id);
	}
	public int getId(Item i)
```

```
	{
		return i.id;
	}
	public String getName()
	{
		return name;
	}
	public double getPrice()
	{
		return price;
	}
	public int getItemStock()
	{
		return stock;
	}
	public abstract String getDetails();
	@Override
	public String toString()
	{
		return getBasicInfo()+" "+getDetails();
	}
	public abstract String getType();
	public abstract void setStock(int s);
	public void setItemStock(int l)
	{
		stock+=l;
	}
}
```

Κλάση Pen :

```
package projectJava;
public class Pen extends Item{
	private String color;
	private double tipSize;
	private static int stockPen=0;
	private static final String Type = "Pen";
	public Pen(String name,double price,String description,int stock,int id,String
color,double tipSize)
	{
		super(name,price,description,stock,id);
		stockPen=stockPen+stock;
		this.color=color;
		this.tipSize=tipSize;
	}
	public String getDetails()
	{
		return "Color: "+color+" Tip size: "+Double.toString(tipSize);
	}
	public static int getStock()
	{
		return stockPen;
```

```
	}
	@Override
	public void setStock(int s)
	{
		stockPen=stockPen+s;
	}
	@Override
	public String getType()
	{
		return Type;
	}
}
```

Κλάση Pencil :

```
package projectJava;
public class Pencil extends Item {
	enum type{
		H,
		B,
		HB
	}
	private double tipSize;
	private type type;
	private static int stockPencil;
	private static final String Type = "Pencil";
	public Pencil(String name,double price,String description,int stock,int
id,double tipSize,type type)
	{
		super(name,price,description,stock,id);
		stockPencil+=stock;
		this.tipSize=tipSize;
		this.type=type;
	}
	public String getDetails()
	{
		return "Tip size: "+Double.toString(tipSize)+" Type: "+type;
	}
	public static int getStock()
	{
		return stockPencil;
	}
	@Override
	public void setStock(int s)
	{
		stockPencil=stockPencil+s;
	}
	@Override
	public String getType()
	{
		return Type;
	}
```

```
}
```

Κλάση Notebook :

```
package projectJava;
public class Notebook extends Item {
	private int sections;
	private static int stockNotebook;
	private static final String Type = "Notebook";
	public Notebook(String name,double price,String description,int stock,int
id,int sections)
	{
		super(name,price,description,stock,id);
		stockNotebook+=stock;
		this.sections=sections;
	}
	public String getDetails()
	{
		return "Sections: "+Integer.toString(sections);
	}
	public static int getStock()
	{
		return stockNotebook;
	}
	@Override
	public String getType()
	{
		return Type;
	}
	@Override
	public void setStock(int s)
	{
		stockNotebook=stockNotebook+s;
	}
}
```

Κλάση Paper :

```
package projectJava;
public class Paper extends Item {
	private int weight;
	private int pages;
	private static int stockPaper;
	private static final String Type = "Paper";
	public Paper(String name,double price,String description,int stock,int id,int
weight,int pages)
	{
		super(name,price,description,stock,id);
		stockPaper+=stock;
```

```
		this.weight=weight;
		this.pages=pages;
	}
	public String getDetails()
	{
		return "Weight: "+Integer.toString(weight)+" Pages:
"+Integer.toString(pages);
	}
	public static int getStock()
	{
		return stockPaper;
	}
	@Override
	public String getType()
	{
		return Type;
	}
	@Override
	public void setStock(int s)
	{
		stockPaper=stockPaper+s;
	}
}
```

Κλάση EShop :

```
package projectJava;
import java.util.*;
public class EShop {
	private static String name;
	private Owner owner;
	private static Item tempItem;
	private static ArrayList<Item> itemsList=new ArrayList<Item>();
	private static ArrayList<Buyer> buyersList=new ArrayList<Buyer>();
	private static ArrayList<String> classList=new ArrayList<String>();
	private static ArrayList<String> itemsNameList=new ArrayList<String>();
	private static ArrayList<String> emailList=new ArrayList<String>();
	private static ArrayList<String> buyerNameList=new ArrayList<String>();
	private static int ProductId;
	private static int error;
	private static Scanner showCat=new Scanner(System.in);
	public EShop(String name,Owner owner)
	{
		this.name=name;
		this.owner=owner;
	}
	public void addItem(Item i)
	{
		do
		{
			error=0;
```

```
		try
		{
			if(itemsList.contains(i))
			{
				throw new CopyException();
			}
			else
			{
				itemsList.add(i);
			}
		}
		catch(CopyException e)
		{
			System.out.println("This item already exists\n");
			error=1;
		}
		}while(error==1);
	}
	public static Item getItemById(int id)
	{
		do
		{
		error=0;
		try
		{
		for(Item i:itemsList)
		{
			if (id==i.getId(i))
			{
		     tempItem=i;
			}
		}
		for(Item i:itemsList)
		{
		if(!(itemsList.contains(i)))
		{
			throw new NoSuchElementException();
		}
		}
		}
		catch(NoSuchElementException e)
		{
			System.out.println("No such element exists. Give another
item\n");
			error=1;
		}
		}while(error==1);
			return tempItem;
	}
	public void removeItem(Item i)
	{
		do
		{
			error=0;
		try
```

```
		{
			if(!(itemsList.contains(i)))
			{
				throw new NoSuchElementException();
			}
			else
			{
				itemsList.remove(i);
			}
		}
		catch(NoSuchElementException e)
		{
			System.out.println("This item doesn't exist, so it can't be
removed\n");
			error=1;
		}
		}while(error==1);
	}
	public static void addBuyer(Buyer b)
	{
		do
		{
			error=0;
		try
		{
			if(buyersList.contains(b))
			{
				throw new CopyException();
			}
			else
			{
				buyersList.add(b);
			}
		}
		catch(CopyException e)
		{
			System.out.println("This buyer already exists\n");
			error=1;
		}
		}while(error==1);
	}
	public static void removeBuyer(Buyer b)
	{
		do
		{
			error=0;
		try
		{
			if(!(buyersList.contains(b)))
			{
				throw new NoSuchElementException();
			}
			else
			{
				buyersList.remove(b);
```

```
                        }
                }
                catch(NoSuchElementException e)
                {
                        System.out.println("This buyer doesn't exist, so he can't be
removed\n");
                        error=1;
                }
                }while(error==1);
        }
        public void NameList()
        {
                for(Item i:itemsList)
                {
                        itemsNameList.add(i.getName());
                }
        }
        public void setClassNameList()
        {
                for(Item i: itemsList)
                {
                        try {
                        String classes =i.getClass().getSimpleName();
                        if(classList.contains(classes))
                        {
                                throw new CopyException();
                        }
                        classList.add(classes);
                        }
                    catch(CopyException e)
                        {
                            classList.remove(i.getClass().getSimpleName());
                        }
                }
        }
        public static void updateItemStock(Item i,int stock)
        {
                        if(i.getType().equals("Pen"))
                        {
                                ((Pen)i).setStock(stock);
                        }
                        else if(i.getType().contentEquals("Pencil"))
                        {
                                ((Pencil)i).setStock(stock);
                        }
                        else if(i.getType().contentEquals("Notebook"))
                        {
                                ((Notebook)i).setStock(stock);
                        }
                        else if(i.getType().contentEquals("Paper"))
                        {
                                ((Paper)i).setStock(stock);
                        }
        }
        public static void showCategories()
```

```
	{
		try
		{
		if(classList.isEmpty())
		{
			throw new NoSuchElementException();
		}
		for(String s:classList)
		{
			if(s.equals("Pen"))
			{
				System.out.println(s+"
"+Integer.toString(Pen.getStock()));
			}
			else if(s.equals("Pencil"))
			{
				System.out.println(s+"
"+Integer.toString(Pencil.getStock()));
			}
			else if(s.equals("Notebook"))
			{
				System.out.println(s+"
"+Integer.toString(Notebook.getStock()));
			}
			else if(s.equals("Paper"))
			{
				System.out.println(s+"
"+Integer.toString(Paper.getStock()));
			}
		}
	    }
		catch(NoSuchElementException e)
		{
			System.out.println("The e-shop is empty\n");
			System.exit(0);
		}
	}
	public static void showProductsInCategory()
	{
		do
		{
			error=0;
			try
			{
		System.out.println("Give the name of the category you wish to see: ");
		String choice=showCat.nextLine();
		if(choice.equals(Pen.class.getSimpleName()))
		{
			int  j=1;
			for (Item i:itemsList)
			{
				if(i instanceof Pen)
				{
					System.out.println(j+" "+i.getName());
					j++;
```

```
                        }
                }
        }
        else if(choice.equals(Pencil.class.getSimpleName()))
        {
                int  j=1;
                for (Item i:itemsList)
                {
                        if(i instanceof Pencil)
                        {
                                System.out.println(j+" "+i.getName());
                                j++;
                        }
                }
        }
        else if(choice.equals(Notebook.class.getSimpleName()))
        {
                int  j=1;
                for (Item i:itemsList)
                {
                        if(i instanceof Notebook)
                        {
                                System.out.println(j+" "+i.getName());
                                j++;
                        }
                }
        }
        else if(choice.equals(Paper.class.getSimpleName()))
        {
                int  j=1;
                for (Item i:itemsList)
                {
                        if(i instanceof Paper)
                        {
                                System.out.println(j+" "+i.getName());
                                j++;
                        }
                }
        }
    else
if((!choice.equals(Paper.class.getSimpleName()))&&(!choice.equals(Notebook.class.getS
impleName()))&&(!choice.equals(Pencil.class.getSimpleName()))&&(!choice.equals(Pen.cl
ass.getSimpleName())))
            {
                 throw new NoSuchElementException();
                }
        }
                catch(NoSuchElementException e)
                {
                        error=1;
                }
                catch(IllegalStateException e)
                {
                        error=1;
                }
```

```
		}while(error==1);
	}
	public static void showProduct()
	{
		System.out.println("Give the name of the product you wish to see: ");
		do
		{
			error=0;
			try
			{
		String choice=showCat.nextLine();
			for(Item i:itemsList)
			{
				if(choice.contains((i.getName())))
				{
					if(i.getItemStock()<=0)
					 {
						System.out.println("This item is out of
stock");
					 }
					else
					{
					System.out.println(i.toString());
					ProductId=i.getId(i);
					}
				}
			}
			if(!(itemsNameList.contains(choice)))
		{
			throw new NoSuchElementException();
		}
			}
			catch(NoSuchElementException e)
			{
				System.out.println("No such element exists\n");
				error=1;
			}
			catch(IllegalStateException e)
			{
				error=1;
			}
		}while(error==1);
	}
	public static int getProductId()
	{
		return ProductId;
	}
	public static void checkStatus()
	{
		int i=1;
		for(Buyer b:buyersList)
		{
			System.out.println(i+". "+b.getUserName()+" "+b.getBonus()+"
"+b.getCategory());
			i++;
```

```
            }
        }
        public static ArrayList<Buyer> getBuyersList()
        {
            return buyersList;
        }
        public static String getEShopName()
        {
            return name;
        }
        public static ArrayList<String> getEmailList()
        {
            for(Buyer b:buyersList)
            {
                emailList.add(b.getEmail());
            }
            return emailList;
        }
        public static ArrayList<String> getBuyerNameList()
        {
            for(Buyer b:buyersList)
            {
                buyerNameList.add(b.getName());
            }
            return buyerNameList;
        }
}
```

Κλάση ItemOrdered :

```
package projectJava;
public class ItemOrdered {
        private Item item;
        private int quantity;
        public ItemOrdered(Item item,int quantity)
        {
            this.item=item;
            this.quantity=quantity;
        }
        public int getQuantity()
        {
            return quantity;
        }
        public Item getItem()
        {
            return item;
        }
        public void setQuantity(int q)
        {
            quantity+=q;
        }
        public String getOrderName()
```

```
	{
		return item.getName();
	}
}
```

Κλάση ShoppingCart :

```
package projectJava;
import java.util.*;
public class ShoppingCart {
	private ArrayList<ItemOrdered> orderList=new ArrayList<ItemOrdered>();
	private ArrayList<String> orderNameList=new ArrayList<String>();
	private static boolean check=false;
	private static int error=0;
	private static Scanner ShoppingCart=new Scanner(System.in);
	public ShoppingCart()
	{

	}
	public void addItemOrdered(Item i,int quantity)
	{
		ItemOrdered order=new ItemOrdered(i,quantity);
		orderList.add(order);
		EShop.updateItemStock(i,-quantity);
		i.setItemStock(-quantity);
	}
	public  void removeItemOrdered(ItemOrdered i)
	{
		int quantity=i.getQuantity();
		if(!check)
		{
		EShop.updateItemStock(i.getItem(),quantity);
		(i.getItem()).setItemStock(quantity);
		}
		else if(check)
		{
		EShop.updateItemStock(i.getItem(),0);
		}
		orderList.remove(i);
		orderNameList.remove(i.getOrderName());
	}
	public void changeItemOrderedQuantity(ItemOrdered i,int quantity)
	{
			if (orderList.contains(i))
			{
				i.setQuantity(quantity);
				EShop.updateItemStock(i.getItem(), -quantity);
			}
	}
	public void showCart(Buyer b)
	{
```

```
		for(ItemOrdered i:orderList)
		{
			System.out.println((i.getItem()).getName()+"
"+(i.getItem()).getPrice());
		}
		System.out.println("Total: " +calculateNet()+ "€ Shipping cost:
"+calculateCourierCost(b)+"€");
	}
	public void clearCart()
	{
		try
		{
			if(orderList.size()>0)
			{
			while(orderList.size()>0)
			{
			removeItemOrdered(orderList.get(0));
			}
			}
			else
			{
				throw new NullPointerException();
			}
		}
		catch(NullPointerException e)
		{
			System.out.println("Your cart is empty\n");
		}
	}
	public void checkout(Buyer b)

	{
		showCart(b);
		System.out.println("Do you wish to proceed to checkout? (1.yes/2.no)");
		do
		{
			error=0;
		try
		{
			int choice=ShoppingCart.nextInt();
		if(choice==1)
		{
			b.awardBonus();
			check=true;
			clearCart();
		}
		else if(choice==2)
		{
			showCart(b);
		}
		else if((choice!=1)&&(choice!=2))
		{
			throw new OneOrTwoException();
		}
		}
```

```
         catch(OneOrTwoException ex)
    {
            System.out.println("The answer must be 1 or 2\n");
       error=1;
    }
       catch(InputMismatchException e)
       {
             System.out.println("You must give a number");
             ShoppingCart.nextLine();
             error=1;
       }
       catch(NoSuchElementException e)
       {
             error=1;
       }
       catch(IllegalStateException e)
       {
             error=1;
       }
       catch(ConcurrentModificationException e)
       {
             error=0;
       }
       }while(error==1);
       check=false;
}
public double calculateNet()
{
       double total=0.0;
       for(ItemOrdered i:orderList)
       {
             total=total+(i.getItem()).getPrice()*i.getQuantity();
       }
       return total;
}
public double calculateCourierCost(Buyer b)
{
       double cost=calculateNet()*0.02;
       String category=b.getCategory();
       switch(category)
       {
             case "BRONZE":
                    {
                          if((cost<3)&&(cost>0))
                          {
                                cost=3;
                          }
                          else if(cost>=3)
                          {
                          cost=cost;
                          }
                          else if(cost==0)
                          {
                            cost=0;
                          }
```

```
                                        break;
                                }
                        case "SILVER":
                        {
                                cost=0.5*cost;
                                break;
                        }
                        case "GOLD":
                        {
                                cost=0;
                                break;
                        }
                }
                return cost;
        }
        public ArrayList<ItemOrdered> getOrderList()
        {
                return orderList;
        }
        public ArrayList<String> getOrderNameList()
        {
                for(ItemOrdered i:orderList)
                {
                        orderNameList.add(i.getOrderName());
                }
                return orderNameList;
        }
        public static void setCheck(boolean b)
        {
                check=b;
        }
}
```

Κλάση Menu :

```
package projectJava;

import java.util.ConcurrentModificationException;
import java.util.InputMismatchException;
import java.util.NoSuchElementException;
import java.util.Scanner;

import projectJava.Buyer.buyerCategory;

public class Menu {
        private Buyer wantedBuyer;
        private static int error=0;
        private ItemOrdered modifiedItem;
        private static Scanner input=new Scanner(System.in);
```

```
	public Menu() throws OneToFiveException,OneOrTwoException,
OneToThreeException, OneToFourException
	{
		emailRequest();
	}
	public void emailRequest() throws OneToFiveException,OneOrTwoException,
OneToThreeException, OneToFourException
	{
		System.out.println("Give your email please: ");
		String email=input.nextLine();
		try
		{
		if(EShop.getEmailList().contains(email))
		{
			for(Buyer b:EShop.getBuyersList())
			{
		if(email.equals(b.getEmail()))
		{
			wantedBuyer=b;
			System.out.println("Welcome " +b.getUserName()+"! Your total
points are: "+b.getBonus()+" and your category is: "+b.getCategory());

	System.out.println("**********************************************************
**************************************************");
			methodForBuyer();
		}
			}
		}
		if(email.equals(Owner.getEmail()))
		{
			    System.out.println("Welcome "+Owner.getName()+"! You are the
owner!");

System.out.println("**********************************************************************
*********************************************");
			    methodForOwner();
		}

	if((!(EShop.getEmailList().contains(email)))&&(!(email.equals(Owner.getEmail()
))))
			   {
			    throw new NoSuchElementException();
			   }
		}
		catch(NoSuchElementException ec)
		{
			System.out.println("This email doesn't exist.\nDo you want to
create a new email account? (1.yes/2.no)\n");
			do
			{
			error=0;
```

```
                    System.out.println("Give a number");
                    try
                    {
                          int choice=input.nextInt();
                    if((choice!=1)&&(choice!=2))
                    {
                          throw new OneOrTwoException();
                    }
                    else if(choice==2)
                    {
                          input.nextLine();
                          emailRequest();
                    }
                    else if(choice==1)
                    {
                          input.nextLine();
                          System.out.println("Give your name\n");
                          String name=input.nextLine();
                          System.out.println("Give an email\n");
                          String newEmail=input.nextLine();
                          ShoppingCart sc=new ShoppingCart();
                          Buyer b=new
Buyer(name,newEmail,0,buyerCategory.BRONZE,sc);
                          EShop.addBuyer(b);
                          emailRequest();
                    }
                    }
                    catch(OneOrTwoException e)
                    {
                          System.out.println("The answer must be 1 or 2\n");
                          error=1;
                    }
                    catch(InputMismatchException e)
                    {
                          input.nextLine();
                          error=1;
                    }
                    }while(error==1);
              }
       }
       public void methodForBuyer() throws OneToFiveException,OneOrTwoException,
OneToThreeException, OneToFourException
       {
              System.out.println(EShop.getEShopName());

       System.out.println("**********************************************************
**************************************************");
           System.out.println("MENU:");
           System.out.println("1.Browse Store ");
           System.out.println("2.View Cart ");
```

```
            System.out.println("3.Checkout ");
            System.out.println("4.Log out ");
            System.out.println("5.Exit ");

System.out.println("**************************************************************************
**********************************************");
            error=0;
            do
            {
               error=0;
            System.out.println("Give a number");
            try
            {
                  int number=input.nextInt();
            if ((number!=1)&&(number!=2)&&(number!=3)&&(number!=4)&&(number!=5))
            {
               throw new OneToFiveException();
            }
            switch(number)
            {
            case 1:
            {
               browseStore();
               break;
            }
            case 2:
            {
               viewCart();
               break;
            }
            case 3:
            {
               Checkout();
               break;
            }
            case 4:
            {
               LogOut();
               break;
            }
            case 5:
            {
               Exit();
               break;
            }
            }
            }
            catch(OneToFiveException ex)
            {
               System.out.println("The number must be from 1 to 5 \n");
```

```
            error=1;
          }
            catch(InputMismatchException e)
            {
                  input.nextLine();
                  error=1;
            }
          }while(error==1);
      }
      public void browseStore() throws OneToFiveException,OneOrTwoException,
OneToThreeException, OneToFourException
      {
            EShop.showCategories();
            EShop.showProductsInCategory();
            EShop.showProduct();
            Item chosenItem=EShop.getItemById(EShop.getProductId());
            if(chosenItem.getItemStock()<=0)
             {
                   browseStore();
             }
            else
            {
            System.out.println("Do you want to add this product to your cart?
(1.yes/2.no)");
            do
            {
                  error=0;
            try
            {
                  int choice=input.nextInt();
          if ((choice!=1)&&(choice!=2))
          {
             throw new OneOrTwoException();
          }
          else if(choice==1)
            {
                  do
                  {
                        error=0;
                        System.out.println("Give the quantity you wish to buy: ");
                        try
                        {
               int quantity=input.nextInt();
                    if((quantity>chosenItem.getItemStock())||(quantity<0))
                    {
                          throw new QuantityException();
                    }
                    try
                    {
```

```
if((wantedBuyer.getCart()).getOrderNameList().contains(chosenItem.getName()))
                        {
                               throw new CopyException();
                        }
                        else
                        {
                    wantedBuyer.placeOrder(chosenItem,quantity);
                         }
                    }
                    catch(CopyException e)
                    {
                           System.out.println("This item already exists, but the
quantity will be updated\n");
                           for(ItemOrdered
io:(wantedBuyer.getCart()).getOrderList())
                           {
                                  if(io.getOrderName().equals(chosenItem.getName()))
                 {
                                        chosenItem.setItemStock(-quantity);

(wantedBuyer.getCart()).changeItemOrderedQuantity(io,quantity);
                 }
                            }
                    }
                         }
                         catch(QuantityException e)
                         {
                                System.out.println("This quantity is not available
\n");
                         error=1;
                         }
                         catch(InputMismatchException e)
                         {
                                System.out.println("You must give a number\n");
                                input.nextLine();
                                error=1;
                         }
                  }while(error==1);
                   System.out.println("\n\nDo you want to continue browsing the
store? (1.yes/2.no)");
                         do
                         {
                                error=0;
                         System.out.println("Give a number");
                         try
                         {
                      int browse=input.nextInt();
                      if ((browse!=1)&&(browse!=2))
                      {
                         throw new OneOrTwoException();
```

```
            }
      if(browse==1)
      {
         browseStore();
      }
      else if(browse==2)
      {
         viewCart();
      }
               }
            catch(OneOrTwoException ex)
            {
               System.out.println("The answer must be 1 or 2\n");
               error=1;
            }
               catch(InputMismatchException e)
               {
                      System.out.println("You must give a number");
                      input.nextLine();
                      error=1;
               }
               }while(error==1);
}
else if(choice==2)
{
       System.out.println("\n\nDo you want to go back? (1.yes/2.no)");
               do
               {
                       error=0;
               System.out.println("Give a number");
               try
               {
            int goBack=input.nextInt();
            if ((goBack!=1)&&(goBack!=2))
            {
               throw new OneOrTwoException();
            }
      if(goBack==1)
      {
            methodForBuyer();
      }
      else if(goBack==2)
      {
         browseStore();
      }
               }
            catch(OneOrTwoException ex)
            {
               System.out.println("The answer must be 1 or 2\n");
               error=1;
```

```
                        }
                           catch(InputMismatchException e)
                           {
                                  System.out.println("You must give a number");
                                  input.nextLine();
                                  error=1;
                           }
                           }while(error==1);
              }
              }
           catch(OneOrTwoException ex)
           {
              System.out.println("The answer must be 1 or 2\n");
              error=1;
           }
              catch(InputMismatchException e)
              {
                     System.out.println("You must give a number");
                     input.nextLine();
                     error=1;
              }
              }while(error==1);
              }
       }
       public void viewCart() throws OneToFiveException,OneOrTwoException,
OneToThreeException, OneToFourException
       {
              (wantedBuyer.getCart()).showCart(wantedBuyer);
              System.out.println("Do you wish to 1.modify an item, 2.clear your cart,
3.go to checkout or 4.go back? ");
              do
              {
                     error=0;
              System.out.println("Give a number");
              try
              {
           int choice=input.nextInt();
           if ((choice!=1)&&(choice!=2)&&(choice!=3)&&(choice!=4))
           {
              throw new OneToFourException();
           }
           switch(choice)
           {
           case 1:
           {
              modifyItem();
            break;
           }
           case 2:
           {
```

```
   System.out.println("Do you want to clear your cart? (1.yes/2.no)");
          do
          {
                error=0;
          System.out.println("Give a number");
          try
          {
       int clearation=input.nextInt();
       if ((clearation!=1)&&(clearation!=2))
       {
          throw new OneOrTwoException();
       }
if(clearation==1)
   {
          (wantedBuyer.getCart()).clearCart();
          for(ItemOrdered i:((wantedBuyer).getCart()).getOrderList())
          {
                (i.getItem()).setItemStock(i.getQuantity());
          }
   }
else if(clearation==2)
{
   viewCart();
}
          }
            catch(OneOrTwoException ex)
       {
                System.out.println("The answer must be 1 or 2\n");
          error=1;
       }
          catch(InputMismatchException e)
          {
                input.nextLine();
                error=1;
          }
          catch(ConcurrentModificationException e)
          {
                error=0;
          }
          }while(error==1);
   System.out.println("\n\nDo you want to go back? (1.yes/2.no)");
          do
          {
                error=0;
          System.out.println("Give a number");
          try
          {
       int gogoBack=input.nextInt();
       if ((gogoBack!=1)&&(gogoBack!=2))
       {
```

```
          throw new OneOrTwoException();
       }
        if(gogoBack==1)
        {
             methodForBuyer();
        }
        else if(gogoBack==2)
        {
          viewCart();
        }
          }
            catch(OneOrTwoException ex)
       {
                System.out.println("The answer must be 1 or 2\n");
          error=1;
       }
          catch(InputMismatchException e)
          {
                input.nextLine();
                error=1;
          }
          }while(error==1);
break;
}
case 3:
{
   Checkout();
   break;
}
case 4:
{
   System.out.println("\n\nDo you want to go back? (1.yes/2.no)");
          do
          {
                error=0;
          System.out.println("Give a number");
          try
          {
       int goBack=input.nextInt();
       if ((goBack!=1)&&(goBack!=2))
       {
          throw new OneOrTwoException();
       }
        if(goBack==1)
        {
             methodForBuyer();
        }
        else if(goBack==2)
        {
          viewCart();
```

```
                 }
                   }
                     catch(OneOrTwoException ex)
                  {
                         System.out.println("The answer must be 1 or 2\n");
                   error=1;
                  }
                   catch(InputMismatchException e)
                   {
                         input.nextLine();
                         error=1;
                   }
                   }while(error==1);
                   break;
            }
            }
               }
               catch(OneToFourException ex)
            {
                   System.out.println("The number must be 1,2,3 or 4 \n");
               error=1;
            }
               catch(InputMismatchException e)
               {
                   input.nextLine();
                   error=1;
               }
               }while(error==1);
            }
            public void modifyItem() throws OneToThreeException, OneToFiveException,
OneToFourException, OneOrTwoException
            {
               if((wantedBuyer.getCart()).getOrderList().isEmpty())
               {
                   System.out.println("Your cart is empty\n");
                   methodForBuyer();
               }
               input.nextLine();
                   System.out.println("Choose the item you wish to modify: ");
                  do
                  {
                  error=0;
                  try
                  {
                     String selected=input.nextLine();
                     if((wantedBuyer.getCart()).getOrderNameList().contains(selected))
                     {
                         for(ItemOrdered i:(wantedBuyer.getCart()).getOrderList())
                         {
                             if(selected.equals((i.getItem()).getName()))
```

```
                                        {
                                        modifiedItem=i;
                                        }
                           }
                           System.out.println("Do you wish to 1.delete it or 2.change
the quantity or 3.go back?");
                           do
                           {
                           error=0;
                           System.out.println("Give a number");
                           try
                           {
                        int modification=input.nextInt();
                        if ((modification!=1)&&(modification!=2)&&(modification!=3))
                        {
                           throw new OneToThreeException();
                        }
                        if(modification==1)
                        {
                           (wantedBuyer.getCart()).removeItemOrdered(modifiedItem);
                           System.out.println("Do you wish to go back?
(1.yes/2.no)");
                           do
                           {
                           error=0;
                           try
                           {
                                  int back=input.nextInt();
                             if(back==1)
                                  {
                                        methodForBuyer();
                                  }
                                  else if(back==2)
                                  {
                                        viewCart();
                                  }
                                  else if((back!=1)&&(back!=2))
                                  {
                                        throw new OneOrTwoException();
                                  }
                                  }
                                   catch(OneOrTwoException ex)
                               {
                                        System.out.println("The answer must be 1 or
2\n");
                                  error=1;
                               }
                           catch(InputMismatchException e)
                                        {
                                  System.out.println("You must give a number");
```

```
                                        input.nextLine();
                                        error=1;
                                    }
                                catch(NoSuchElementException e)
                                {
                                    error=1;
                                }
                                catch(ConcurrentModificationException e)
                                {
                                    error=0;
                                }
                                }while(error==1);
                        }
                        else if(modification==2)
                        {
                           System.out.println("Do you want to 1.increase or
2.decrease the quantity or 3.go back? ");
                                        do
                                        {
                                        error=0;
                                        System.out.println("Give a number");
                                        try
                                        {
                                    int changed=input.nextInt();
                                    if ((changed!=1)&&(changed!=2)&&(changed!=3))
                                    {
                                       throw new OneToThreeException();
                                    }
                                    input.nextLine();
                                System.out.println("Type the quantity you wish to
add/substract: ");
                                do
                                {
                                   error=0;
                                try
                                {
                                int amountation=input.nextInt();

if(((amountation>0)&&(amountation>(modifiedItem.getItem()).getItemStock()))||((amount
ation<0)&&(Math.abs(amountation)>modifiedItem.getQuantity())))
                                {
                                   throw new QuantityException();
                                }
                                else
                                {
                                if(changed==1)
                            {
                                   (modifiedItem.getItem()).setItemStock(-amountation);

      (wantedBuyer.getCart()).changeItemOrderedQuantity(modifiedItem,amountation);
```

```
                                System.out.println("Item modified");
                                viewCart();
                            }
                                else if(changed==2)
                            {
                                (modifiedItem.getItem()).setItemStock(-amountation);

       (wantedBuyer.getCart()).changeItemOrderedQuantity(modifiedItem,amountation);
                                System.out.println("Item modified");
                                viewCart();
                            }
                                else if(changed==3)
                                {
                                modifyItem();
                                }
                                }
                                }
                                catch(QuantityException e)
                                {
                                System.out.println("Not available quantity, please
type a new one");
                                error=1;
                                }
                                }while(error==1);
                        }
                                catch(OneToThreeException ex)
                                    {
                                        System.out.println("The number must be 1,2
or 3 \n");
                                      error=1;
                                    }
                                catch(InputMismatchException e)
                                      {
                                      System.out.println("You must give a number");
                                             input.nextLine();
                                             error=1;
                                      }
                                      }while(error==1);
                                      }

                            else if(modification==3)
                            {
                              modifyItem();
                            }
                    }
                              catch(OneToThreeException ex)
                            {
                                      System.out.println("The number must be 1,2 or 3
\n");
                              error=1;
```

```
                        }
                   catch(InputMismatchException e)
                               {
                          System.out.println("You must give a number");
                                      input.nextLine();
                                      error=1;
                               }
                          }while(error==1);
                }
                   else
                   {
                          throw new NoSuchElementException();
                   }
                }
                catch(NoSuchElementException e)
                {
                   System.out.println("This item doesn't exist in your order list,
please give another item\n");
                   error=1;
                }
           }while(error==1);
           }
           public void Checkout() throws OneToFiveException,OneOrTwoException,
OneToThreeException, OneToFourException
           {
              System.out.println("The cost of your order is:
"+Double.toString((wantedBuyer.getCart()).calculateNet())+" and the shipping cost is:
"+Double.toString((wantedBuyer.getCart()).calculateCourierCost(wantedBuyer))+".");
              (wantedBuyer.getCart()).checkout(wantedBuyer);
              System.out.println("\n\nDo you want to go back? (1.yes/2.no)");
                     do
                     {
                            error=0;
                     System.out.println("Give a number");
                     try
                     {
                 int goBack=input.nextInt();
                 if ((goBack!=1)&&(goBack!=2))
                 {
                    throw new OneOrTwoException();
                 }
                  if(goBack==1)
                  {
                       methodForBuyer();
                  }
                  else if(goBack==2)
                  {
                 System.out.println("Do you want to 1.logout or 2.go to the
menu?\n");
                     do
```

```
                  {
                        error=0;
                  System.out.println("Give a number");
                  try
                  {
               int menu=input.nextInt();
               if ((menu!=1)&&(menu!=2))
               {
                  throw new OneOrTwoException();
               }
               else if(menu==1)
               {
                  LogOut();
               }
               else if(menu==2)
               {
                  methodForBuyer();
               }
                  }
                  catch(OneOrTwoException ex)
               {
                        System.out.println("The answer must be 1 or 2\n");
                  error=1;
               }
                  catch(InputMismatchException e)
                              {
                                    input.nextLine();
                                    error=1;
                              }
                  }while(error==1);
                }
                  }
               catch(OneOrTwoException ex)
               {
                  System.out.println("The answer must be 1 or 2\n");
                  error=1;
               }
                  catch(InputMismatchException e)
                        {
                              input.nextLine();
                              error=1;
                        }
                  }while(error==1);
          }
          public void LogOut() throws OneToFiveException,OneOrTwoException,
OneToThreeException, OneToFourException
          {
             System.out.println("Are you sure you want to logout? (1.yes/2.no)");
                  do
                  {
```

```
                        error=0;
                   System.out.println("Give a number");
                   try
                   {
               int logout=input.nextInt();
               if ((logout!=1)&&(logout!=2))
               {
                  throw new OneOrTwoException();
               }
               if(logout==1)
               {
                  input.nextLine();
                  emailRequest();
               }
               if(logout==2)
               {
                  methodForBuyer();
               }
                  }
               catch(OneOrTwoException ex)
               {
                  System.out.println("The answer must be 1 or 2\n");
                  error=1;
               }
                  catch(InputMismatchException e)
                        {
                              input.nextLine();
                              error=1;
                        }
                  }while(error==1);
          }
          public void Exit() throws OneToFiveException,OneOrTwoException,
OneToThreeException, OneToFourException
          {
             System.out.println("Are you sure you want to exit? (1.yes/2.no)");
                   do
                   {
                         error=0;
                   System.out.println("Give a number");
                   try
                   {
               int exit=input.nextInt();
               if ((exit!=1)&&(exit!=2))
               {
                  throw new OneOrTwoException();
               }
               if(exit==1)
               {
                  System.exit(0);
               }
```

```
                if(exit==2)
                {
                   methodForBuyer();
                }
                   }
                catch(OneOrTwoException ex)
                {
                   System.out.println("The answer must be 1 or 2\n");
                   error=1;
                }
                   catch(InputMismatchException e)
                          {
                                 input.nextLine();
                                 error=1;
                          }
                   }while(error==1);
          }
          public void methodForOwner() throws OneToFiveException,OneOrTwoException,
OneToThreeException,OneToFourException
          {
             System.out.println(EShop.getEShopName());

      System.out.println("************************************************************
**************************************************");
             System.out.println("MENU:");
                 System.out.println("1.Browse Store ");
                 System.out.println("2.Check Status ");
                 System.out.println("3.Log out ");
                 System.out.println("4.Exit ");

System.out.println("********************************************************************
**********************************************");
                do
                {
                   error=0;
                System.out.println("Give a number");
                try
                {
                    int number=input.nextInt();
                if ((number!=1)&&(number!=2)&&(number!=3)&&(number!=4))
                {
                   throw new OneToFourException();
                }
                switch(number)
                {
                case 1:
                {
                   browseStoreOwner();
                   break;
                }
```

```
                case 2:
                {
                   CheckStatus();
                   break;
                }
                case 3:
                {
                   LogOutOwner();
                   break;
                }
                case 4:
                {
                   ExitOwner();
                   break;
                }
                }
                }
                catch(OneToFourException ex)
                {
                   System.out.println("The number must be 1,2,3 or 4 \n");
                   error=1;
                }
               catch(InputMismatchException e)
                         {
                                input.nextLine();
                                error=1;
                         }
                }while(error==1);

          }
          public void browseStoreOwner() throws
OneToFiveException,OneOrTwoException, OneToThreeException, OneToFourException
          {
                   EShop.showCategories();
                   EShop.showProductsInCategory();
                   EShop.showProduct();
                   System.out.println("Do you want to modify this product?
(1.yes/2.no)");
                   do
                   {
                          error=0;
                   System.out.println("Give a number");
                   try
                   {
                int choice=input.nextInt();
                if ((choice!=1)&&(choice!=2))
                {
                   throw new OneOrTwoException();
                }
                   if(choice==1)
```

```
                    {
                            System.out.println("Give the quantity you wish to modify:
(if you want to reduce the quantity of the selected item, please give a negative
number) ");
                            do
                            {
                                  error=0;
                                  System.out.println("Give a number\n");
                            try
                            {
                        int quantity=input.nextInt();
                            Item chosenItem=EShop.getItemById(EShop.getProductId());


if((quantity<0)&&(Math.abs(quantity)>chosenItem.getItemStock()))
                            {
                                  throw new QuantityException();
                            }
                            else
                            {
                                  EShop.updateItemStock(chosenItem,quantity);
                                  chosenItem.setItemStock(quantity);
                            }
                            System.out.println("\n\nDo you want to continue browsing
the store? (1.yes/2.no)");
                           do
                           {
                                 error=0;
                           System.out.println("Give a number");
                           try
                           {
                        int browse=input.nextInt();
                        if ((browse!=1)&&(browse!=2))
                        {
                           throw new OneOrTwoException();
                        }
                         if(browse==1)
                         {
                           browseStoreOwner();
                         }
                         else if(browse==2)
                         {
                           methodForOwner();
                         }
                           }
                        catch(OneOrTwoException ex)
                        {
                           System.out.println("The answer must be 1 or 2\n");
                           error=1;
                        }
```

```
                        catch(InputMismatchException e)
                              {
                                    input.nextLine();
                                    error=1;
                              }
                        }while(error==1);
                        }
                        catch(QuantityException e)
                        {
                              System.out.println("This quantity is not available
\n");
                        error=1;
                        }
                        catch(InputMismatchException e)
                        {
                              input.nextLine();
                              error=1;
                        }
                  }while(error==1);
                  }
                  else if(choice==2)
                  {
                        System.out.println("\n\nDo you want to go back?
(1.yes/2.no)");
                        do
                        {
                              error=0;
                        System.out.println("Give a number");
                        try
                        {
                    int goBack=input.nextInt();
                    if ((goBack!=1)&&(goBack!=2))
                    {
                       throw new OneOrTwoException();
                    }
                     if(goBack==1)
                     {
                          methodForOwner();
                     }
                     else if(goBack==2)
                     {
                        browseStoreOwner();
                     }
                      }
                    catch(OneOrTwoException ex)
                    {
                      System.out.println("The answer must be 1 or 2\n");
                      error=1;
                    }
                      catch(InputMismatchException e)
```

```
                                {
                                        input.nextLine();
                                        error=1;
                                }
                        }while(error==1);
                   }
                   }
                catch(OneOrTwoException ex)
                {
                   System.out.println("The answer must be 1 or 2\n");
                   error=1;
                }
                   catch(InputMismatchException e)
                        {
                                input.nextLine();
                                error=1;
                        }
                   }while(error==1);
           }
           public void CheckStatus() throws OneToFiveException,OneOrTwoException,
OneToThreeException, OneToFourException
           {
              EShop.checkStatus();
              System.out.println("Do you wish to 1.modify a buyer or 2.go back?");
                           do
                           {
                                   error=0;
                           System.out.println("Give a number");
                           try
                           {
                       int moded=input.nextInt();
                       if ((moded!=1)&&(moded!=2))
                       {
                          throw new OneOrTwoException();
                       }
              if(moded==1)
              {
                     if(EShop.getBuyersList().isEmpty())
                           {
                                   System.out.println("The buyer list is empty\n");
                                   methodForOwner();
                           }
                     input.nextLine();
              System.out.println("Give the name of the buyer you wish to see: ");
              do
              {
                     error=0;
                     try
                     {
                     String choice=input.nextLine();
```

```
                    if(EShop.getBuyerNameList().contains(choice))
                    {
                    for(Buyer b:EShop.getBuyersList())
                    {
                           if(choice.equals(b.getUserName()))
                           {
                                  wantedBuyer=b;
                                  (wantedBuyer.getCart()).showCart(wantedBuyer);
                                  System.out.println("Do you wish to delete this buyer
from the buyers list? (1.yes/2.no)");
                                  do
                                  {
                                         error=0;
                                  System.out.println("Give a number");
                                  try
                                  {
                               int deleted=input.nextInt();
                               if ((deleted!=1)&&(deleted!=2))
                               {
                                  throw new OneOrTwoException();
                               }
                                  if(deleted==1)
                                  {
                                         EShop.removeBuyer(wantedBuyer);
                                         ShoppingCart.setCheck(false);
                                         (wantedBuyer.getCart()).clearCart();
                                         CheckStatus();
                                  }
                                  else if(deleted==2)
                                  {
                                         CheckStatus();
                                  }
                                  }
                               catch(OneOrTwoException ex)
                               {
                                  System.out.println("The answer must be 1 or 2\n");
                                  error=1;
                               }
                                  catch(InputMismatchException e)
                                         {
                                                input.nextLine();
                                                error=1;
                                         }
                                  }while(error==1);
                           }
                    }
                    }
                    else if(!(EShop.getBuyerNameList().contains(choice)))
                    {
                           throw new NoSuchElementException();
```

```
                    }
                    }
                    catch(NoSuchElementException e)
                    {
                       System.out.println("This buyer does not exist, please type
another name\n");
                           error=1;
                    }
              }while(error==1);
              }
              else if(moded==2)
                    {
                           methodForOwner();
                    }
                           }
                        catch(OneOrTwoException ex)
                        {
                           System.out.println("The answer must be 1 or 2\n");
                           error=1;
                        }
                           catch(InputMismatchException e)
                                 {
                                        input.nextLine();
                                        error=1;
                                 }
                           }while(error==1);
           }
          public void LogOutOwner() throws OneToFiveException,OneOrTwoException,
OneToThreeException, OneToFourException
           {
              System.out.println("Are you sure you want to logout? (1.yes/2.no)");
                           do
                           {
                                 error=0;
                           System.out.println("Give a number");
                           try
                           {
                        int logout=input.nextInt();
                        if ((logout!=1)&&(logout!=2))
                        {
                           throw new OneOrTwoException();
                        }
                    if(logout==1)
                    {
                       input.nextLine();
                       emailRequest();
                    }
                    if(logout==2)
                    {
                       methodForOwner();
```

```
                }
                        }
                      catch(OneOrTwoException ex)
                      {
                        System.out.println("The answer must be 1 or 2\n");
                        error=1;
                      }
                        catch(InputMismatchException e)
                              {
                                    input.nextLine();
                                    error=1;
                              }
                        }while(error==1);
          }
         public void ExitOwner() throws OneToFiveException,OneOrTwoException,
OneToThreeException, OneToFourException
          {
             System.out.println("Are you sure you want to exit? (1.yes/2.no)");
                   do
                   {
                         error=0;
                   System.out.println("Give a number");
                   try
                   {
                int exit=input.nextInt();
                if ((exit!=1)&&(exit!=2))
                {
                   throw new OneOrTwoException();
                }
                if(exit==1)
                {
                  System.exit(0);
                }
                if(exit==2)
                {
                  methodForOwner();
                }
                   }
                catch(OneOrTwoException ex)
                {
                  System.out.println("The answer must be 1 or 2\n");
                  error=1;
                }
                  catch(InputMismatchException e)
                        {
                              input.nextLine();
                              error=1;
                        }
                  }while(error==1);
          }
```

```
}
```

Κλάση Main :

```
package projectJava;

import projectJava.Pencil.type;

import projectJava.Buyer.buyerCategory;

public class Main {

	public static void main(String args[]) throws
OneToFiveException,OneOrTwoException, OneToThreeException, OneToFourException,
CopyException

	{

		Owner owner=new Owner("Paul","zeopeo@hotmail.gr",true);

		EShop eshop=new EShop("That's a wrap!",owner);

		Pen p1=new Pen("Bic",0.5,"ball-point",23,80934,"blue",0.6);

		Pen p2=new Pen("Pilot",0.7,"ball-point",14,98723,"red",0.5);

		Pen p3=new Pen("Parker",0.8,"ball-point",22,98463,"green",0.4);

		Pencil pc1=new Pencil("Staedtler",0.9,"draw pencil",34,10987,0.3,
type.HB);

		Pencil pc2 = new Pencil("Palomino",1.5,"Graphite
pencil",58,52178,0.4,type.B);

		Pencil pc3 = new Pencil("Blackwing",3.2,"Solid graphite
pencil",12,14127,0.5,type.H);

		Notebook n1=new Notebook("SKAG",3.5,"binder notebook",25,78907,3);

		Notebook n2=new Notebook("GLOBUS",5.0,"A4 notebook",45,34569,4);

		Notebook n3=new Notebook("Jumbo",2.5,"binder notebook",75,78906,2);

		Paper pp1=new Paper("Xerox",5.0,"A4 yellow",35,13245,10,1000);

		Paper pp2=new Paper("Papex",6.5,"A3 white",30,87654,15,600);

	Paper pp3=new Paper("Origamix",7.0,"A2 white",65,54673,10,750);

	eshop.addItem(p1);

	eshop.addItem(p2);

	eshop.addItem(p3);

	eshop.addItem(pc1);

	eshop.addItem(pc2);
```

```
            eshop.addItem(pc3);
            eshop.addItem(n1);
            eshop.addItem(n2);
            eshop.addItem(n3);
            eshop.addItem(pp1);
            eshop.addItem(pp2);
            eshop.addItem(pp3);
            eshop.NameList();
            eshop.setClassNameList();
            ShoppingCart sc1=new ShoppingCart();
            ShoppingCart sc2=new ShoppingCart();
            Buyer b1=new Buyer("Eva","eva@gmail.com",200,buyerCategory.GOLD,sc1);
            eshop.addBuyer(b1);
            Buyer b2=new
Buyer("Demetres","jimmy@yahoo.de",60,buyerCategory.BRONZE,sc2);
            eshop.addBuyer(b2);
            b1.placeOrder(p1, 3);
            b1.placeOrder(p3,4);
            b1.placeOrder(n2, 2);
            b1.placeOrder(pp1,1);
            b2.placeOrder(p2, 2);
            b2.placeOrder(n1, 2);
            b2.placeOrder(n3, 3);
            b2.placeOrder(pp2,1);
            Menu m=new Menu();
      }
}
```

Κλάσεις Εξαιρέσεων :

CopyException :

```
package projectJava;

public class CopyException extends Exception{
public CopyException()
{
}
}
```

QuantityException :

```
package projectJava;

public class QuantityException extends Exception{
	public QuantityException()
	{
	}
}
```

OneOrTwoException :

```
package projectJava;

public class OneOrTwoException extends Exception{
public OneOrTwoException()
{
}
}
```

OneToThreeException :

```
package projectJava;

public class OneToThreeException extends Exception{
public OneToThreeException()
{
}
}
```

OneToFourException :

```
package projectJava;
```

```
public class OneToFourException extends Exception{
public OneToFourException()
{
}
}
```

OneToFiveException :

```
package projectJava;

public class OneToFiveException extends Exception{
public OneToFiveException()
{
}
}
```