**PROJECT ΟΝΤΟΚΕΝΤΡΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ**

**ΜΕΛΗ ΟΜΑΔΑΣ: ΑΜ: ΕΤΟΣ: EMAIL:**

**ΚΑΡΑΪΣΚΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ 1072636 1Ο st1072636@ceid.upatras.gr**

**ΠΑΠΑΓΕΩΡΓΙΟΥ ΔΗΜΗΤΡΙΟΣ 1064280 3Ο up1064280@upatras.gr**

Το project έχει υλοποιηθεί χρησιμοποιώντας το περιβάλλον : IntelliJ IDEA.

Ο κώδικας έχει αποθηκευτεί στον Google Drive, και παρατίθεται και στο τέλος των αναφορών. Το link είναι: <https://drive.google.com/drive/folders/1F193c39tOLXqut6_5xt27CjWJXTGDkur?usp=sharing>

Τα διαγράμματα των κλάσεων περιέχονται σε αρχείο του κώδικα.

Παρακάτω ακολουθεί σύντομη περιγραφή των περιεχομένων της κάθε κλάσης .

CLASS USER(ABSTRACT)

Εκτός από τις private μεταβλητές name, email, προστέθηκαν και οι μέθοδοι:

* setUserName(String Name) : για την αρχικοποίηση του ονόματος ενός χρήστη
* setUserEmail(String newEmail): για την αρχικοποίηση του email ενός χρήστη
* getUserName(): επιστρέφει το όνομα του αντικειμένου-χρήστη
* getUserEmail(): επιστρέφει το email του αντικειμένου-χρήστη
* getUserDetails(): abstract μέθοδος που επικαλύπτεται από τις Buyer και Owner

CLASS OWNER

Εκτός από την private Boolean isAdmin = true, προστέθηκαν ο constructor:

* Owner(String newName): ο οποίος αρχικοποιεί το όνομα ενός νέου αντικειμένου Owner

Και η μέθοδος:

* getUserDetails(): η οποία επικαλύπτει αυτή που ορίστηκε στην κλάση user και επιστρέφει την τιμή της isAdmin

CLASS BUYER

Εκτός από τις μεταβλητές bonus, buyerCategory, cart και τις μεθόδους awardBonus(), setBuyerCategory(), placeOrder(), προστέθηκαν ο constructor:

* Buyer(String Name): ο οποίος αρχικοποιεί το όνομα ενός αντικειμένου τύπου Buyer

Και οι μέθοδοι:

* removeOrder(Eshop obj, String itemsName): η οποία αφαιρεί ένα αντικείμενο από την λίστα αγορών του εκάστοτε χρήστη, καλώντας την αντίστοιχη μέθοδο της ShoppingCart
* changeQuantity(Eshop obj, String itemsName, int newAmount): αλλάζει την ποσότητα ενός αντικειμένου από τη λίστα του αγοραστή μέσω μεθόδου της ShoppingCart
* clearBuyersCart(Eshop obj) : αδειάζει το καλάθι του αγοραστή μέσω μεθόδου της ShoppingCart
* checkOutBuyer(): καλεί την αντίστοιχη μέθοδο της ShoppingCart
* getUserDetails(): επιστρέφει τα στοιχεία του User που την καλεί
* getBuyerCategory(): επιστρέφει την κατηγορία του αγοραστή
* getBuyerCart(): εμφανίζει το καλάθι του αγοραστή και επιστρέφει “empty”/”not empty”, ανάλογα

Και οι δοσμένες μέθοδοι:

* awardBonus(double orderCost): δέχεται σαν όρισμα το κόστος της παραγγελίας και καλεί την setBuyerCategory()
* setBuyerCategory(int points): μεταβάλει την κατηγορία του πελάτη ανάλογα με τους πόντους του
* placeOrder(EShop obj, String itemsName): ζητά από τον χρήστη να εισάγει το όνομα του προϊόντος που θέλει να προσθέσει στη λίστα του, δημιουργεί ένα αντικείμενο ItemOrdered και καλεί την addItemOrdered()

CLASS ITEM (ABSTRACT)

Εκτός από τις μεταβλητές name, price, description, stock, id και τις μεθόδους getBasicInfo(), getDetails() και toString(), έχουν προστεθεί και οι μέθοδοι:

* setItem(String Name): η οποία παίρνει σαν όρισμα το όνομα του νέου προϊόντος, το θέτει, και εμφανίζει μηνύματα για την αρχικοποίηση της τιμής και του stock του
* setName(String Name): αρχικοποιεί το όνομα ενός προϊόντος
* setID(int newID): αρχικοποιεί το ID ενός προϊόντος
* setDescription(String newDescription): αρχικοποιεί την περιγραφή ενός προϊόντος
* setStock(int newStock): αρχικοποιεί/μεταβάλει την ποσότητα ενός προϊόντος
* getName(): επιστρέφει το όνομα του προϊόντος
* getDescription(): επιστρέφει την περιγραφή του προϊόντος
* getStock(): επιστρέφει το stock του προϊόντος
* getPrice(): επιστρέφει την τιμή του προϊόντος
* (abstract) setMoreItemInfo(): η οποία επικαλύπτεται στις υποκλάσεις pen, pencil, notebook, paper

Οι δοσμένες μέθοδοι:

* getBasicInfo(): επιστρέφει τα στοιχεία του προϊόντος
* (abstract) getDetails(): επικαλύπτεται στις υποκλάσεις pen, pencil, paper, notebook
* toString(): επιστρέφει τις μεθόδους getBasicInfo() και getDetails()

CLASS PEN

Εκτός από τις μεταβλητές color, tipSize και την επικαλυπτόμενη μέθοδο getDetails(), έχουν προστεθεί ο constructor:

* Pen(String name): που αρχικοποιεί το όνομα ενός αντικειμένου τύπου pen

Και η μέθοδος:

* setMoreItemInfo(): μέσω της οποίας αρχικοποιούνται το χρώμα και η tipSize

Η δοσμένη επικαλυπτόμενη μέθοδος:

* getDetails(): επιστρέφει το περιεχόμενο της color και της tipSize του αντικειμένου τύπου Pen

CLASS PENCIL

Εκτός από τις μεταβλητές tipSize, type και την επικαλυπτόμενη μέθοδο getDetails(), έχουν προστεθεί και ο constructor:

* Pencil(String Name): αρχικοποιεί το όνομα ενός νέου αντικειμένου τύπου pencil

Και η μέθοδος:

* setMoreItemInfo(): μέσω της οποίας αρχικοποιούνται η tipSize και η type

Η δοσμένη επικαλυπτόμενη μέθοδος:

* getDetails(): επιστρέφει το περιεχόμενο της tipSize και της type του αντικειμένου τύπου Pencil

CLASS NOTEBOOK

Εκτός από την μεταβλητή sections και την επικαλυπτόμενη μέθοδο getDetails(), έχουν προστεθεί ο constructor:

* NoteBook(String Name): αρχικοποιεί το όνομα ενός νέου αντικειμένου notebook

Και η μέθοδος:

* setMoreItemInfo(): μέσω της οποίας αρχικοποιείται η sections

Και η δοσμένη μέθοδος:

* getDetails(): επιστρέφει το περιεχόμενο της sections

CLASS PAPER

Εκτός από τις μεταβλητές weight, pages και την επικαλυπτόμενη μέθοδο getDetails(), έχουν προστεθεί ο constructor:

* Paper(String Name): αρχικοποιεί το όνομα ενός νέου αντικειμένου τύπου paper

Και η μέθοδος:

* setMoreItemInfo(): μέσω της οποίας αρχικοποιούνται οι weight και pages

Και η επικαλυπτόμενη μέθοδος:

* getDetails(): επιστρέφει το περιεχόμενο της weight και της pages

CLASS ESHOP

Εκτός από τις μεταβλητές name, owner, itemsList, buyersList, έχουν προστεθεί οι μέθοδοι:

* removeBuyer(String buyerForRemove): διαγράφει τον buyer με όνομα ίδιο με αυτό του ορίσματος. Επιστρέφει “Buyer removed” αν το όνομα υπήρχε στη λίστα. Αλλιώς επιστρέφει “The buyer’s name you want to remove is not in list”.

Οι ακόλουθες μέθοδοι είναι παρόμοιες με κάποιες από τις δοσμένες, προσαρμοσμένες κατάλληλα για την κλάση ShoppingCart, και αφορούν την λίστα των προϊόντων του καταστήματος.

* lookFor(String item, int quantity): ελέγχει αν υπάρχει αντικείμενο με ίδιο όνομα με αυτό στο όρισμα, σε ίδια ή λιγότερη ποσότητα και επιστρέφει έναν αριθμό ανάλογα με το αποτέλεσμα της αναζήτησης
* alterStock(String itemsName, int amount): μεταβάλλει την ποσότητα ενός αντικειμένου στη λίστα τον προϊόντων του καταστήματος
* getSelectedItemsPrice(String itemsName): επιστρέφει την τιμή του αντικειμένου με το όνομα του ορίσματος

Οι ακόλουθες μέθοδοι έχουν δημιουργηθεί για την κλάση MENU κατά το log-in και σε λειτουργία του μενού για χρήστη “owner”.

* checkEmail(String emailLog): ελέγχει την ύπαρξη user με email αντίστοιχο με αυτού του ορίσματος και επιστρέφει true/false ανάλογα
* getUser(String e\_mail): επιστρέφει τον user με το αντίστοιχο email
* getNumberOfCustomers(): επιστρέφει τον αριθμό των πελατών στην buyersList της Eshop

Οι δοσμένες μέθοδοι:

* addItem(): ρωτά τον χρήστη για την κατηγορία του προϊόντος που θέλει να δημιουργήσει. Έπειτα ζητά από το χρήστη να εισάγει το όνομα του προϊόντος, αν υπάρχει ήδη, εγείρεται εξαίρεση “item already in list!”. Εάν όχι δημιουργείται αντικείμενο αντίστοιχου τύπου με την προσαχθείσα κατηγορία και ζητά από το χρήστη την αρχικοποίηση των υπολοίπων στοιχείων του
* getItemById(): ζητά από το χρήστη να εισάγει έναν αριθμό, από την αριθμημένη λίστα των προϊόντων. Εάν αυτός ο αριθμός αντιστοιχεί σε κάποιο αντικείμενο, εμφανίζονται τα στοιχεία του, αλλιώς εγείρεται εξαίρεση “ID not in list”
* removeItem(): ζητά από τον χρήστη να εισάγει το όνομα του αντικειμένου που θέλει να αφαιρέσει από την λίστα των προϊόντων. Εάν δεν υπάρχει εμφανίζεται ανάλογο μήνυμα
* addBuyer(): εάν η buyersList είναι άδεια, ζητείται από τον χρήστη να καταχωρήσει έναν owner και τα στοιχεία του. Αλλιώς ζητά τα στοιχεία του πελάτη για καταχώρηση. Και στις δύο περιπτώσεις, εάν υπάρχει ήδη ο owner/buyer εγείρεται εξαίρεση “owner already exist in list” ή “buyer already exist in buyers list” αντίστοιχα
* updateItemStock(String product): ενημερώνει το stock του αντίστοιχου προϊόντος, αλλιώς εμφανίζει το μήνυμα “The item is not in list.”
* showCategories(): εμφανίζει τις κατηγορίες των προϊόντων και των αριθμό των προϊόντων που υπάρχουν σε κάθε κατηγορία
* showProductsInCategory(String ans): εμφανίζει αριθμημένη λίστα με τα προϊόντα της κατηγορίας που δόθηκε σαν όρισμα. Επιστρέφει “ok” ή “not ok”, ανάλογα με το αν η κατηγορία που δόθηκε υπάρχει ή όχι
* showProduct(): ζητά από τον χρήστη να εισάγει το όνομα του προϊόντος για το οποίο θέλει να δει λεπτομέρειες, και επιστρέφει τις λεπτομέρειές του. Εάν η συγκεκριμένη κατηγορία είναι κενή επιστρέφει “empty\_ category”, αλλιώς επιστρέφει “not in list” εάν δεν υπάρχει το αντικείμενο στην κατηγορία
* checkStatus(): εμφανίζει αριθμημένη λίστα με τους πελάτες του eshop

CLASS ITEMORDERED

Εκτός από τις μεταβλητές name, cost και quantity έχουν προστεθεί οι constructors:

* ItemOrdered(String Name, int amount, double price): αρχικοποιεί το όνομα, την ποσότητα και την τιμή του προϊόντος που προστίθεται στην λίστα αγορών ενός πελάτη
* ItemOrdered(String Name, int amount): αρχικοποιεί το όνομα και την ποσότητα ενός προϊόντος που προστίθεται στην λίστα αγορών ενός πελάτη

Και οι μέθοδοι:

* setQuantity(int wantedQuantity): μεταβάλει την ποσότητα ενός προϊόντος, που υπάρχει ήδη στο καλάθι του πελάτη
* setCost(double price): θέτει την τιμή του προϊόντος που την καλεί ίση με αυτή του ορίσματός της
* ItemsName(): επιστρέφει το όνομα του αντικειμένου ItemOrdered που την καλεί
* getQuantity(): επιστρέφει την ποσότητα του αντικειμένου ItemOrdered που την καλεί
* getItemsPrice(): επιστρέφει το κόστος του αντικειμένου ItemOrdered που την καλεί

΄

CLASS SHOPPINGCART

Δεν έχει γίνει κάποια επιπλέον προσθήκη στην παρούσα κλάση, επομένως, οι δοσμένες μέθοδοι:

* addItemOrdered(Eshop obj, String itemsName, ItemOrdered tempItem): αρχικά ελέγχει, μέσω της μεθόδου lookFor(), για την ύπαρξη του προϊόντος, που δόθηκε ως όρισμα, στην λίστα της Eshop, καθώς και για την απαιτούμενη ποσότητα. Εάν δεν υπάρχει κάτι από τα δύο εγείρονται οι εξαιρέσεις “The stock of this product is less than the desired quantity!” και “The product you are looking for is not in the product list!” αντίστοιχα. Εάν δεν κληθεί κάποια από αυτές τις εξαιρέσεις, εκτελεί τη λειτουργία που αναγράφεται στην εκφώνηση
* removeItemOrdered(Eshop obj, String itemForRemove): διαγράφει από τη λίστα αγορών του χρήστη το δοσμένο ως όρισμα προϊόν και ενημερώνει το stock του καταστήματος
* changeItemOrderedQuantity(Eshop obj, String answer, int newAmount): αρχικά ελέγχει εάν υπάρχει η ζητούμενη ποσότητα, του προϊόντος , στο κατάστημα. Εάν υπάρχει προσθέτει στην ποσότητα του προϊόντος της λίστας του πελάτη την επιπλέον ζητούμενη ποσότητα και ενημερώνει το stock της Eshop. Αλλιώς εγείρεται εξαίρεση “The stock of this product is less than the desired quantity!”
* showCart(): εάν το καλάθι του πελάτη είναι άδειο εμφανίζει αντίστοιχο μήνυμα και επιστρέφει “empty”, αλλιώς εμφανίζει αριθμημένη λίστα με τα προϊόντα στο καλάθι του, το συνολικό κόστος και επιστρέφει “not empty”
* clearCart(Eshop obj): καλεί την removeItemOrdered() για κάθε αντικείμενο στο καλάθι
* checkout(Buyer currBuyer): η λειτουργία της μεθόδου είναι αυτή που αναγράφεται στην εκφώνηση
* calculateNet(): υπολογίζει και επιστρέφει το κόστος της παραγγελίας
* calculateCurrierCost(Buyer CurrBuyer): υπολογίζει και επιστρέφει το κόστος των μεταφορικών εξόδων της παραγγελίας, με βάση τις παραμέτρους που τίθενται στην εκφώνηση

CLASS MENU

Όλες οι μέθοδοι, εκτός της LogIn (public) που αποτελεί την βασική μέθοδο, είναι private και χρησιμοποιούνται μέσω των menuOptions(), για users, και ownerOptions(), για owners, και αυτές οι δυο μέσω της LogIn() :

* LogIn(Eshop eShop): χρησιμοποιεί μία μεταβλητή , continue\_using, για να εμφανίζει το μήνυμα για είσοδο χρήστη, έως ότου να γίνει ψευδής και να τερματίσει το πρόγραμμα. Εάν το email που εισαχθεί αντιστοιχεί σε owner καλείται η ownerOptions(), εάν αντιστοιχεί σε user καλείται η menuOptions() και αν δεν αντιστοιχεί σε κανέναν γίνεται ερώτηση για εγγραφή νέου χρήστη.
* menuOptions(Buyer customer, EShop eshop): εμφανίζει χαιρετισμό και τα στοιχεία του χρήστη. Έπειτα μια αριθμημένη λίστα με τις επιλογές του χρήστη, όπου με είσοδο κάποιας εκ των επιλογών, καλείται η αντίστοιχη μέθοδος
* BroseStore(Buyer customer, EShop eshop): εμφανίζεται αριθμημένη λίστα με τις κατηγορίες των προϊόντων του καταστήματος, με κλήση της αντίστοιχης μεθόδου της Eshop, και ρωτά τον χρήστη την κατηγορία και το αντίστοιχο προϊόν της κατηγορίας για το οποίο θέλει να δει λεπτομέρειες
* ViewCart(EShop eshop, Buyer buyer): εμφανίζει αριθμημένη λίστα του καλαθιού του πελάτη και αντίστοιχη λίστα με τις ενέργειες που αναγράφονται στην εκφώνηση
* CheckOut(Buyer buyer): καλείται η αντίστοιχη μέθοδος της κλάσης Buyer
* Back(int optionNum): επιστρέφει τον αριθμό την προηγούμενης από την παρούσα ενέργεια, μεταβιβάζοντας τον χρήστη ένα επίπεδο πιο πάνω στο menu
* LogOut(): ρωτά τον χρήστη εάν επιθυμεί να συνδεθεί με άλλο λογαριασμό. Εάν ναι, τον επιστρέφει στην LogIn(), αλλιώς τερματίζει το πρόγραμμα
* Exit(): τερματίζει το πρόγραμμα
* ownerOptions(Owner owner, EShop shop): εμφανίζει χαιρετισμό στο χρήστη, τα στοιχεία του καθώς και την επιβεβαίωση ότι είναι Administrator, και το αριθμημένο menu επιλογών του
* ownerBroseStore(Owner owner, EShop shop): η λειτουργία της είναι αυτή που αναγράφεται στην εκφώνηση
* CheckStatus(EShop shop): αρχικά, εάν η λίστα των πελατών εμφανίζει αντίστοιχο μήνυμα. Αλλιώς εμφανίζει αριθμημένη λίστα με τους πελάτες και τις επιλογές για προβολή των στοιχείων κάποιου πελάτη και του καλαθιού του, καθώς και την δυνατότητα διαγραφής του από την λίστα, με κατάλληλη ενημέρωση του stock εάν εκκρεμεί παραγγελία στο όνομά του

**ΠΑΡΑΚΑΤΩ ΑΝΑΓΡΑΦΕΤΑΙ Ο ΚΩΔΙΚΑΣ ΚΑΘΕ ΚΛΑΣΗΣ**

**CLASS USER**

import java.util.Scanner;  
  
public abstract class User {  
  
 private String name;  
 private String email;  
  
 Scanner keyboard = new Scanner(System.*in*);  
   
 //methods  
 public void setUserName(String Name)  
 {  
 name = Name;  
 }  
  
 public void setUserEmail(String newEmail)  
 {  
 email = newEmail;  
 }  
  
 public abstract String getUserDetails();  
  
 public String getUserName()  
 {  
 return name;  
 }  
  
 publiC String getUsersEmail() { return email; }  
}

**CLASS BUYER**

public class Buyer extends User {  
 private int bonus=0;  
 private String buyerCategory = "Bronze"; // Bronze,Silver,Gold  
 private ShoppingCart cart = new ShoppingCart();  
  
  
 //constructor  
  
 public Buyer(String Name)  
 {  
 this.setUserName(Name);  
 }  
  
 //methods  
  
 public void awardBonus(double orderCost)  
 {  
 double bonusCalc = orderCost\*0.1;  
 this.bonus = (int)Math.*floor*(bonusCalc);  
 setBuyerCategory(this.bonus);  
 }  
  
 private void setBuyerCategory(int points)  
 {  
 if(points> 100 && points<=200 )  
 this.buyerCategory = "Silver"; // Silver 50% στα μεταφορικά  
 else if(points > 200)  
 this.buyerCategory = "Gold"; // Gold χωρις μεταφορικά  
 }  
  
 public void placeOrder(EShop obj,String itemsName)  
 {  
 System.*out*.println("Enter the desired quantity of the item you want: ");  
 int desiredQuantity = keyboard.nextInt();  
 String nchar = keyboard.nextLine();  
  
 ItemOrdered tempItem = new ItemOrdered(itemsName,desiredQuantity);  
 cart.addItemOrdered(obj,itemsName,tempItem);  
  
 }  
  
 public void removeOrder(EShop obj, String itemsName)  
 {  
 cart.removeItemOrdered(obj,itemsName);  
 }  
  
 public void changeQuantity(EShop obj,String itemsName,int newAmount)  
 {  
 cart.changeItemOrderedQuantity(obj,itemsName,newAmount);  
 }  
  
 public void clearBuyersCart(EShop obj)  
 {  
 cart.clearCart(obj);  
 }  
  
 public boolean checkOutBuyer()  
 {  
 return cart.checkOut(this);  
 }  
  
 public String getUserDetails()  
 {  
 return "Name: " + this.getUserName() + ", points: " + this.bonus + ", category: " + this.buyerCategory + ", email: " + this.getUsersEmail();  
 }  
  
 public String getBuyerCategory(){ return this.buyerCategory; }  
  
 public String getBuyerCart()  
 {  
 String cart\_state = this.cart.showCart();  
 if(cart\_state.equalsIgnoreCase("empty"))  
 {  
 return "empty";  
 }  
 else  
 return "not empty";  
 }  
}

**CLASS OWNER**

public class Owner extends User{  
 private Boolean isAdmin = true;  
  
 //constructors  
  
 public Owner(String newName)  
 {  
 this.setUserName(newName);  
 }  
  
 //methods  
  
 public String getUserDetails()  
 {  
 return isAdmin.toString();  
 }  
}

**CLASS ITEM**

import java.util.Scanner;  
  
public abstract class Item {  
 public String name;  
 private double price;  
 private String description;  
 private int stock;  
 private int id;  
  
 Scanner keyboard = new Scanner(System.*in*);  
  
 //methods  
  
 public void setItem(String Name)  
 {  
 name = Name;  
 System.*out*.println("Enter item's price:");  
 price = keyboard.nextDouble();  
 String nChar = keyboard.nextLine();  
 System.*out*.println("Enter item's stock:");  
 stock = keyboard.nextInt();  
 }  
  
 public void setName(String Name)  
 {  
 name = Name;  
 }  
  
 public void setID(int newID)  
 {  
 id = newID;  
 }  
  
 public void setDescription(String newDescription)  
 {  
 description = newDescription;  
 }  
  
 public void setStock(int newStock)  
 {  
 stock = stock + newStock;  
 }  
  
 public String getName()  
 {  
 return name;  
 }  
  
 public String getDescription()  
 {  
 return description;  
 }  
  
 public int getStock() { return stock; }  
  
 public double getPrice() { return price; }  
  
 public String getBasicInfo()  
 {  
 return "Name: " + this.name + ", Price: " + this.price + "€ , Stock: " + this.stock +  
 ", ID: " + this.id;  
 }  
  
 public String toString()  
 {  
 getBasicInfo();  
 getDetails();  
 return getBasicInfo() + ", " + getDetails();  
 }  
   
 //abstracts  
 public abstract String getDetails();  
   
 public abstract void setMoreItemInfo();  
}

**CLASS PEN**

import java.util.Scanner;  
  
public class Pen extends Item {  
 private String color;  
 private double tipSize; // in mm  
  
 Scanner keyboard = new Scanner(System.*in*);  
   
 //constructor  
 public Pen(String Name)  
 {  
 name = Name;  
 }  
   
 //methods  
 public String getDetails()  
 {  
 return "color: " + this.color + ", tip size: " + this.tipSize;  
 }  
  
 public void setMoreItemInfo()  
 {  
 System.*out*.println("Enter pen's color:");  
 color = keyboard.nextLine();  
 System.*out*.println("Enter pen's tipSize:");  
 tipSize = keyboard.nextDouble();  
 String nchar = keyboard.nextLine();  
 }  
}

**CLASS PENCIL**

import java.util.Scanner;  
  
public class Pencil extends Item{  
 private double tipSize; // in mm  
 private String type;   
  
 Scanner keyboard = new Scanner(System.*in*);  
  
 //constructor  
 public Pencil(String Name)  
 {  
 name = Name;  
 }  
  
 //methods  
 public String getDetails()  
 {  
 return " Tip size " + this.tipSize + ", Type " + this.type;  
 }  
  
 public void setMoreItemInfo()  
 {  
 System.*out*.println("Enter pencil's tipSize:");  
 tipSize = keyboard.nextDouble();  
 String nchar = keyboard.nextLine();  
 System.*out*.println("Enter pencil's type(H,B,HB):");  
 type = keyboard.nextLine();  
 }  
}

**CLASS PAPER**

import java.util.Scanner;  
  
public class Paper extends Item{  
 private int weight; // in grams  
 private int pages;  
  
 Scanner keyboard = new Scanner(System.*in*);  
  
 //constructor  
 public Paper(String Name)  
 {  
 name = Name;  
 }  
  
 //methods  
 public String getDetails()  
 {  
 return " Weight: " + this.weight + "gr Pages: " + this.pages;  
 }  
  
 public void setMoreItemInfo()  
 {  
 System.*out*.println("Enter paper's weight:");  
 weight = keyboard.nextInt();  
 String Nchar = keyboard.nextLine();  
 System.*out*.println("Enter paper's pages:");  
 pages = keyboard.nextInt();  
 String nchar = keyboard.nextLine();  
 }  
}

**CLASS NOTEBOOK**

import java.util.Scanner;  
  
public class NoteBook extends Item {  
 private int sections; // αριθμος θεμάτων  
  
 Scanner keyboard = new Scanner(System.*in*);  
  
 //constructor  
 public NoteBook(String Name)  
 {  
 name = Name;  
 }  
  
 //methods  
 public String getDetails()  
 {  
 return " Sections: " + this.sections;  
 }  
  
 public void setMoreItemInfo()  
 {  
 System.*out*.println("Enter the number of sections in the notebook");  
 sections = keyboard.nextInt();  
 String nchar = keyboard.nextLine();  
 }  
}

**CLASS ESHOP**

import java.util.ArrayList;  
import java.util.Scanner;  
  
public class EShop {  
 private String name;  
 private Owner owner;  
 private ArrayList<Item> itemsList = new ArrayList<>();  
 private ArrayList<User> buyersList = new ArrayList<>();  
  
 Scanner keyboard = new Scanner(System.*in*);  
  
 //methods  
  
 //ΠΡΟΣΘΗΚΗ ΠΡΟΙΟΝΤΩΝ  
 public void addItem() //μπορεί να χρειαστεί προσθήκες  
 {  
 try {  
 System.*out*.println("Enter the product's category which you want to add:");  
 String category = keyboard.nextLine();  
  
 System.*out*.println("Enter the name of the item you want to add in items list.");  
 String newItem = keyboard.nextLine();  
  
  
 for(int i=0; i<itemsList.size(); i++)  
 {  
 if(newItem.equalsIgnoreCase(itemsList.get(i).getName()))  
 throw new Exception("Exception: item already in list!");  
 }  
  
  
 if(category.equalsIgnoreCase("Pen")) {  
 itemsList.add(new Pen(newItem));  
 itemsList.get(itemsList.size()-1).setDescription("Pen");  
 itemsList.get(itemsList.size()-1).setItem(newItem);  
 itemsList.get(itemsList.size()-1).setID(itemsList.size()-1);  
 itemsList.get(itemsList.size()-1).setMoreItemInfo();  
 }  
 else if(category.equalsIgnoreCase("Pencil")) {  
 itemsList.add(new Pencil(newItem));  
 itemsList.get(itemsList.size()-1).setDescription("Pencil");  
 itemsList.get(itemsList.size()-1).setItem(newItem);  
 itemsList.get(itemsList.size()-1).setID(itemsList.size()-1);  
 itemsList.get(itemsList.size()-1).setMoreItemInfo();  
 } else if(category.equalsIgnoreCase("Notebook")) {  
 itemsList.add(new NoteBook(newItem));  
 itemsList.get(itemsList.size()-1).setDescription("NoteBook");  
 itemsList.get(itemsList.size()-1).setItem(newItem);  
 itemsList.get(itemsList.size()-1).setID(itemsList.size()-1);  
 itemsList.get(itemsList.size()-1).setMoreItemInfo();  
 } else if(category.equalsIgnoreCase("Paper")) {  
 itemsList.add(new Paper(newItem));  
 itemsList.get(itemsList.size()-1).setDescription("Paper");  
 itemsList.get(itemsList.size()-1).setItem(newItem);  
 itemsList.get(itemsList.size()-1).setID(itemsList.size()-1);  
 itemsList.get(itemsList.size()-1).setMoreItemInfo();  
 } else {  
 System.*out*.println("No such category in our products!");  
 }  
  
  
 }  
 catch(Exception e)  
 {  
 System.*out*.println(e.getMessage());  
 }  
 }  
  
 public void getItemById()  
 {  
 try {  
 System.*out*.println("Enter the item number you want to find.");  
 int itemID = keyboard.nextInt();  
 String nchar = keyboard.nextLine();  
 if(itemID<0 || itemID>=itemsList.size())  
 throw new Exception("Exception: ID not in list.");  
 else  
 System.*out*.println(itemsList.get(itemID).toString());  
 }  
 catch(Exception e)  
 {  
 System.*out*.println(e.getMessage());  
 }  
  
 }  
  
 public void removeItem()  
 {  
 boolean isOnList = false;  
 System.*out*.println("Enter the item you want to remove from list.");  
 String itemForRemove = keyboard.nextLine();  
  
 for(int i=0; i<itemsList.size(); i++)  
 {  
 if(itemForRemove.equalsIgnoreCase(itemsList.get(i).getName()))  
 {  
 itemsList.remove(i);  
 isOnList = true;  
 break;  
 }  
 }  
  
 if(!isOnList)  
 System.*out*.println("The item you want to remove is not in list.");  
  
 }  
  
 public void addBuyer()  
 {  
 try  
 {  
 if(buyersList.size()==0)  
 {  
 System.*out*.println("Register an owner for the shop.");  
 System.*out*.print("Enter the name of the owner to add: ");  
 String newOwner = keyboard.nextLine();  
  
 for (int i = 0; i < buyersList.size(); i++) {  
 if (newOwner.equalsIgnoreCase(buyersList.get(i).getUserName()))  
 throw new Exception("Exception: owner already exists in list.");  
 }  
  
 owner = new Owner(newOwner);  
 buyersList.add(owner);  
 for(User obj: buyersList)  
 {  
 if(obj.getUserName().equalsIgnoreCase(newOwner))  
 {  
 System.*out*.print("Enter email address: ");  
 String eAddress = keyboard.nextLine();  
 obj.setUserEmail(eAddress);  
 break;  
 }  
 }  
 }  
 else {  
 System.*out*.print("Enter the name of the buyer to add in customer list: ");  
 String newBuyer = keyboard.nextLine();  
  
 for (int i = 0; i < buyersList.size(); i++) {  
 if (newBuyer.equalsIgnoreCase(buyersList.get(i).getUserName()))  
 throw new Exception("Exception: buyer already exists in buyers list.");  
 }  
  
 buyersList.add(new Buyer(newBuyer));  
 for(User obj: buyersList)  
 {  
 if (obj.getUserName().equalsIgnoreCase(newBuyer))  
 {  
 System.*out*.print("Enter email address: ");  
 String eAddress = keyboard.nextLine();  
 obj.setUserEmail(eAddress);  
 }  
 }  
 }  
 }  
 catch(Exception e)  
 {  
 System.*out*.println(e.getMessage());  
 }  
 }  
  
 public String removeBuyer(String buyerForRemove)  
 {  
 for(int i=0; i<buyersList.size(); i++)  
 {  
 if(buyerForRemove.equalsIgnoreCase(buyersList.get(i).getUserName()))  
 {  
 buyersList.remove(i);  
 return "Buyer removed.";  
 }  
 }  
  
 return "The buyer's name you want to remove is not in list";  
 }  
  
  
 //Ανανεωση ποσοτητας ενος προιντος στο καταστημα  
 public void updateItemStock(String product)  
 {  
 boolean itemNotFound = true;  
  
 for(int i=0; i<itemsList.size(); i++)  
 {  
 if(product.equalsIgnoreCase(itemsList.get(i).getName()))  
 {  
 System.*out*.println("Enter number of units to add in current stock.");  
 int newStock = keyboard.nextInt();  
 String nchar = keyboard.nextLine();  
 itemsList.get(i).setStock(newStock);  
 itemNotFound = false;  
 break;  
 }  
 }  
  
 if(itemNotFound)  
 System.*out*.println("The item is not in list.");  
 }  
  
 //Δειχνει τις κατηγοριες //Λογικα δεν θα χρεαιστει καποια αλλη προσθηκη  
 public void showCategories()  
 {  
 int penItems=0, pencilItems=0, noteBookItems=0, paperItems=0;  
  
 for(Item obj: itemsList)  
 {  
 if(obj.getDescription().equalsIgnoreCase("Pen"))  
 penItems++;  
 else if(obj.getDescription().equalsIgnoreCase("Pencil"))  
 pencilItems++;  
 else if(obj.getDescription().equalsIgnoreCase("NoteBook"))  
 noteBookItems++;  
 else  
 paperItems++;  
 }  
  
 System.*out*.println("The categories of our products are:");  
 System.*out*.println("1." + " Pens" + "(" + penItems + ")");  
 System.*out*.println("2." + " Pencils" + "(" + pencilItems + ")");  
 System.*out*.println("3." + " NoteBooks" + "(" + noteBookItems + ")");  
 System.*out*.println("4." + " Paper" + "(" + paperItems + ")");  
 }  
  
 //Δειχνει τα προιοντα ανα κατηγορια  
 public String showProductsInCategory(String ans)  
 {  
 if(ans.equalsIgnoreCase("Pens"))  
 {  
 int i = 1;  
 for(Item obj: itemsList)  
 {  
 if(obj.getDescription().equalsIgnoreCase("pen")) {  
 System.*out*.println(i + ". " + obj.getName());  
 i++;  
 }  
 }  
 return "ok";  
 }  
 else if(ans.equalsIgnoreCase("pencils"))  
 {  
 int i = 1;  
 for(Item obj: itemsList)  
 {  
 if(obj.getDescription().equalsIgnoreCase("pencil")) {  
 System.*out*.println(i + ". " + obj.getName());  
 i++;  
 }  
 }  
 return "ok";  
 }  
 else if(ans.equalsIgnoreCase("notebooks"))  
 {  
 int i = 1;  
 for(Item obj: itemsList)  
 {  
 if(obj.getDescription().equalsIgnoreCase("notebook")) {  
 System.*out*.println(i + ". " + obj.getName());  
 i++;  
 }  
 }  
 return "ok";  
 }  
 else if(ans.equalsIgnoreCase("paper"))  
 {  
 int i = 1;  
 for(Item obj: itemsList)  
 {  
 if(obj.getDescription().equalsIgnoreCase("paper")) {  
 System.*out*.println(i + ". " + obj.getName());  
 i++;  
 }  
 }  
 return "ok";  
 }  
 else {  
 return "not ok";  
 }  
 }  
  
 public String showProduct()  
 {  
 System.*out*.println("For which product would you like to see more details?");  
 String ans = keyboard.nextLine();  
  
 int i;  
  
 for(i=0; i<itemsList.size(); i++)  
 {  
 if(ans.equalsIgnoreCase(itemsList.get(i).getName()))  
 {  
 System.*out*.println(itemsList.get(i).toString());  
 return itemsList.get(i).getName();  
 }  
 }  
  
 if(i == itemsList.size()) { return "empty\_category"; }  
  
 return "not in list";  
 }  
 public void checkStatus()  
 {  
 int i = 1;  
  
 for (User obj : buyersList)  
 {  
 if (!(obj.getUserDetails().equalsIgnoreCase("true"))) {  
 System.*out*.println(i + ". " + obj.getUserDetails());  
 i++;  
 }  
 }  
  
 }  
   
   
 //adjustedForShoppingCartMethods  
 public int lookFor(String item,int quantity)  
 {  
 for(int i=0; i<this.itemsList.size(); i++)  
 {  
 if(item.equalsIgnoreCase(this.itemsList.get(i).getName()))  
 {  
 if (quantity <= this.itemsList.get(i).getStock())  
 return 0;  
 else  
 return 1;  
 }  
 }  
  
 return 2;  
 }  
  
 public void alterStock(String itemsName,int amount)  
 {  
  
 for(int i=0; i<itemsList.size(); i++)  
 {  
 if(itemsName.equalsIgnoreCase(itemsList.get(i).getName()))  
 {  
 itemsList.get(i).setStock(-amount);  
 break;  
 }  
 }  
 }  
  
 public double getSelectedItemsPrice(String itemsName)  
 {  
 for(Item obj: itemsList)  
 {  
 if(itemsName.equalsIgnoreCase(obj.getName()))  
 return obj.getPrice();  
 }  
  
 return 0;  
 }  
  
 //FOR MENU CLASS, LOG IN  
 public boolean CheckEmail(String emailLog)  
 {  
 for(int i=0; i<buyersList.size(); i++)  
 {  
 if (emailLog.equalsIgnoreCase(buyersList.get(i).getUsersEmail()))  
 {  
 return true;  
 }  
 }  
  
 return false;  
 }  
  
 //o buyer kappa δεν εμφανίζεται ποτέ, αφού η μέθοδος χρησιμοποιείται μόνο αφού έχει πιστοποιηθεί το email  
 public User getUser(String e\_mail)  
 {  
 Buyer kappa = new Buyer("kappa");  
  
 for(int i=0; i<buyersList.size(); i++)  
 {  
 if (e\_mail.equalsIgnoreCase(buyersList.get(i).getUsersEmail()))  
 {  
 return buyersList.get(i);  
 }  
 }  
 return kappa;  
 }  
   
 public int getNumberOfCustomers()  
 {  
 int customer\_count = 0;  
 for (User obj : buyersList)  
 {  
 if (!(obj.getUserDetails().equalsIgnoreCase("true"))) {  
 customer\_count++;  
 }  
 }  
  
 return customer\_count;  
 }  
}

**CLASS ITEMORDERED**

public class ItemOrdered {  
 private String name;  
 private double cost;  
 private int quantity;  
  
 //constructors  
  
 public ItemOrdered(String Name,int amount,double price)  
 {  
 name = Name;  
 quantity = amount;  
 cost = price;  
 }  
  
 public ItemOrdered(String Name, int amount)  
 {  
 name = Name;  
 quantity = amount;  
 }  
   
 //methods  
 public void setQuantity(int wantedQuantity)  
 {  
 quantity = wantedQuantity + this.quantity;  
 }  
 public void setCost(double price) { cost = price; }  
 public String ItemsName()  
 {  
 return name;  
 }  
 public int getQuantity() { return quantity; }  
 public double getItemsPrice() { return cost; }  
}

**CLASS SHOPPINGCART**

import java.util.ArrayList;  
import java.util.Scanner;  
  
public class ShoppingCart{  
 private ArrayList<ItemOrdered> orderList = new ArrayList<>(10);  
  
 Scanner keyboard = new Scanner(System.*in*);  
 //methods  
  
 public void addItemOrdered(EShop obj,String itemsName,ItemOrdered tempItem) //νομιζω οτι ειναι κομπλε  
 {  
  
 try  
 {  
 if(obj.lookFor(itemsName, tempItem.getQuantity()) == 1)  
 throw new Exception("The stock of this product is less than the desired quantity!");  
 else if(obj.lookFor(itemsName, tempItem.getQuantity()) == 2)  
 throw new Exception("The product you are looking for is not in the product list!");  
  
 int i\_counter=0;  
 if(this.orderList.size()==0)  
 {  
 this.orderList.add(new ItemOrdered(tempItem.ItemsName(),tempItem.getQuantity(),obj.getSelectedItemsPrice(tempItem.ItemsName())));  
 obj.alterStock(itemsName,tempItem.getQuantity());  
 }  
 else  
 {  
 for(int i =0; i<this.orderList.size(); i++)  
 {  
 if(itemsName.equalsIgnoreCase(this.orderList.get(i).ItemsName()))  
 {  
 this.orderList.get(i).setQuantity(tempItem.getQuantity());  
 obj.alterStock(itemsName,tempItem.getQuantity());  
 break;  
 }  
 else  
 i\_counter++;  
 }  
  
 if(i\_counter == this.orderList.size())  
 {  
 this.orderList.add(new ItemOrdered(tempItem.ItemsName(),tempItem.getQuantity(),obj.getSelectedItemsPrice((tempItem.ItemsName()))));  
 obj.alterStock(itemsName,tempItem.getQuantity());  
 }  
 }  
  
 }  
 catch(Exception e)  
 {  
 System.*out*.println(e.toString());  
 }  
  
   
 }  
  
 public void removeItemOrdered(EShop obj,String itemForRemove)  
 {  
 for(int i=0; i<this.orderList.size(); i++)  
 {  
 if(itemForRemove.equalsIgnoreCase(this.orderList.get(i).ItemsName()))  
 {  
 int returnedAmount = this.orderList.get(i).getQuantity();  
 obj.alterStock(itemForRemove,-returnedAmount);  
 this.orderList.remove(i);  
 break;  
 }  
 }  
 }  
  
 public void changeItemOrderedQuantity(EShop obj,String answer,int newAmount)  
 {  
 for(int i=0; i<this.orderList.size(); i++)  
 {  
 if(answer.equalsIgnoreCase(this.orderList.get(i).ItemsName()))  
 {  
 try  
 {  
 if (obj.lookFor(answer, newAmount) == 1)  
 throw new Exception("The stock of this product is less than the desired quantity!");  
 else  
 {  
 this.orderList.get(i).setQuantity(newAmount);  
 obj.alterStock(answer,newAmount);  
 }  
 }  
 catch (Exception e)  
 {  
 System.*out*.println(e.toString());  
 }  
 }  
 }  
 }  
  
 public String showCart()  
 {  
 if(this.orderList.size()==0)  
 {  
 System.*out*.println("You cart is empty.");  
 return "empty";  
 }  
 else {  
 for (int i = 0; i < this.orderList.size(); i++) {  
 System.*out*.println(i+1 + ".Item: " + this.orderList.get(i).ItemsName() + ", Quantity: " + this.orderList.get(i).getQuantity() + ", Price: " + this.orderList.get(i).getItemsPrice()+ "€");  
  
 }  
 System.*out*.println("The total cost of your order is: " + calculateNet() + "€") ;  
 return "not empty";  
 }  
 }  
  
 public void clearCart(EShop obj)  
 {  
 for(int i=orderList.size()-1; i>=0; i--)  
 {  
 removeItemOrdered(obj,orderList.get(i).ItemsName());  
 }  
 }  
  
 public boolean checkOut(Buyer currBuyer)  
 {  
 System.*out*.println("You cart contains: ");  
 showCart();  
  
 System.*out*.println("Are you sure you want to check out?(y/n)");  
 String checkOut = keyboard.nextLine();  
  
 if(checkOut.equalsIgnoreCase("y"))  
 {  
 if (currBuyer.getBuyerCategory().equalsIgnoreCase("Gold")) {  
 System.*out*.println("Order's total cost is: " + calculateNet() + "(Category: " + currBuyer.getBuyerCategory() + ")");  
 System.*out*.println("Thank you for visiting us.");  
 currBuyer.awardBonus(calculateNet());  
 this.orderList.clear();  
 return true;  
 } else if (currBuyer.getBuyerCategory().equalsIgnoreCase("silver"))  
 {  
 if(calculateCurrierCost(currBuyer) == 1)  
 {  
 System.*out*.print("Add more items in your cart to get them delivered.");  
 return false;  
 }  
 else {  
 System.*out*.println("Order's total cost is: " + (calculateNet() + calculateCurrierCost(currBuyer) / 2) + "(Category: " + currBuyer.getBuyerCategory() + ")");  
 System.*out*.println("Thank you for visiting us.");  
 currBuyer.awardBonus(calculateNet());  
 this.orderList.clear();  
 return true;  
 }  
 } else {  
 if(calculateCurrierCost(currBuyer) == 1)  
 {  
 System.*out*.println("Order's cost is too low to deliver it.");  
 return false;  
 }  
 else {  
 System.*out*.println("Order's total cost is: " + (calculateNet() + calculateCurrierCost(currBuyer)) + " (Category: " + currBuyer.getBuyerCategory() + ")");  
 System.*out*.println("Thank you for visiting us.");  
 currBuyer.awardBonus(calculateNet());  
 this.orderList.clear();  
 return true;  
 }  
 }  
  
 }  
 return false;  
 }  
  
  
 public double calculateNet()  
 {  
 double totalCost=0;  
 for(int i=0; i<orderList.size(); i++)  
 {  
 totalCost += orderList.get(i).getQuantity() \* orderList.get(i).getItemsPrice();  
 }  
  
 return totalCost;  
 }  
  
 public double calculateCurrierCost(Buyer currBuyer)  
 {  
 double currierCost = 0.2 \* this.calculateNet();  
  
 if(currBuyer.getBuyerCategory().equalsIgnoreCase("gold"))  
 {  
 return 0;  
 }  
  
 if(currierCost < 3)  
 {  
 System.*out*.println("The total cost of your order must be more than 3 € in order to deliver it to you.");  
 return 1;  
 }  
 else if(currBuyer.getBuyerCategory().equalsIgnoreCase("silver"))  
 {  
 return currierCost/2;  
 }  
 else if(currBuyer.getBuyerCategory().equalsIgnoreCase("bronze"))  
 {  
 return currierCost;  
 }  
  
 return -1;  
 }  
   
}

**CLASS MENU**

import java.util.Scanner;  
  
public class Menu {  
 Scanner keyboard = new Scanner(System.*in*);  
  
 public void LogIn(EShop eShop)  
 {  
 boolean continue\_using = true;  
 while(continue\_using) {  
 System.*out*.println("Enter your email to connect: ");  
 String emailLog = keyboard.nextLine();  
  
  
 if (eShop.CheckEmail(emailLog))  
 {  
 if (eShop.getUser(emailLog).getUserDetails().equalsIgnoreCase("true"))  
 {  
 ownerOptions((Owner)eShop.getUser(emailLog), eShop);  
 } else {  
 menuOptions((Buyer)eShop.getUser(emailLog), eShop);  
 }  
 } else {  
 System.*out*.print("The email you entered does not match a buyer/owner in our list." + '\n' + "Would you like to add this user in buyers list?(y/n)");  
 String ans = keyboard.nextLine();  
 if (ans.equalsIgnoreCase("y")) {  
 eShop.addBuyer();  
 System.*out*.println("Registration was successful.");  
 }  
 else  
 {  
 System.*out*.println("Exit or continue using?(exit/continue)");  
 String yeahOrNah = keyboard.nextLine();  
 if(yeahOrNah.equalsIgnoreCase("exit"))  
 continue\_using = false;  
 }  
 }  
 }  
 }  
  
 private void menuOptions(Buyer customer,EShop eShop)  
 {  
 System.*out*.println("Hello: " + customer.getUserName() + "! Welcome to our EShop.");  
 System.*out*.println(customer.getUserDetails());  
  
 System.*out*.println("Menu options: ");  
 System.*out*.println("1. Browse store" + '\n' + "2. View cart" + '\n' + "3. Check Out" + '\n' + "4. Back" +  
 '\n' + "5. Log out" + '\n' + "6. Exit");  
 System.*out*.println("----------------------------------------------");  
  
  
 System.*out*.print("Enter a number between 1-6 to choose an option: ");  
 int optionNum = keyboard.nextInt();  
 String nchar = keyboard.nextLine();  
 System.*out*.println("----------------------------------------------");  
  
 boolean exit = false;  
 while(!exit)  
 {  
  
 while(optionNum<1 || optionNum>6)  
 {  
 System.*out*.println("Enter a number between 1-6 to proceed.");  
 System.*out*.print("Enter the number of the option you want to continue with: ");  
 int newNumber = keyboard.nextInt();  
 String Nchar = keyboard.nextLine();  
 optionNum = newNumber;  
 }  
  
 switch (optionNum)  
 {  
 case(1):  
 {  
 BrowseStore(customer,eShop);  
 System.*out*.print("Would you like to go back or exit?(back/exit/no)" +  
 "\n(Warning! : choosing back from this point will return you to log-in screen.) ");  
 String back\_exit = keyboard.nextLine();  
 if(back\_exit.equalsIgnoreCase("back"))  
 {  
 exit = true;  
 LogOut();  
 }  
 else if(back\_exit.equalsIgnoreCase("exit"))  
 {  
 Exit();  
 }  
 else  
 {  
 System.*out*.print("Enter the option's number you want to continue with: ");  
 optionNum = keyboard.nextInt();  
 break;  
 }  
 break;  
 }  
 case(2):  
 {  
 boolean done\_notDone = ViewCart(eShop,customer);  
 if(done\_notDone)  
 {  
 exit = true;  
 nchar = keyboard.nextLine();  
 break;  
 }  
 else {  
 System.*out*.print("Would you like to go back or exit?(back/exit/no)");  
 String back\_exit = keyboard.nextLine();  
 if (back\_exit.equalsIgnoreCase("back")) {  
 optionNum = Back(optionNum);  
 break;  
 } else if (back\_exit.equalsIgnoreCase("exit")) {  
 Exit();  
 } else {  
 System.*out*.print("Enter the option's number you want to continue with: ");  
 optionNum = keyboard.nextInt();  
 break;  
 }  
 }  
  
 break;  
 }  
 case(3):  
 {  
 boolean done\_not = CheckOut(customer);  
 if(done\_not)  
 {  
 exit = true;  
 nchar = keyboard.nextLine();  
 break;  
 }  
 else {  
 String charr = keyboard.nextLine();  
 System.*out*.print("Would you like to go back or exit?(back/exit/no)");  
 String back\_exit = keyboard.nextLine();  
 if (back\_exit.equalsIgnoreCase("back")) {  
 optionNum = Back(optionNum);  
 break;  
 } else if (back\_exit.equalsIgnoreCase("exit")) {  
 Exit();  
 } else {  
 System.*out*.println("\n" + "------------------------------------");  
 eShop.showCategories();  
 System.*out*.print("Enter the option's number you want to continue with: ");  
 optionNum = keyboard.nextInt();  
 nchar = keyboard.nextLine();  
 break;  
 }  
 }  
 break;  
 }  
 case(4):  
 {  
 optionNum = Back(optionNum);  
 break;  
 }  
 case(5):  
 {  
 String nullChar = keyboard.nextLine();  
 LogOut();  
 exit = true;  
 break;  
 }  
 case(6):  
 {  
 Exit();  
 break;  
 }  
 }  
  
 }  
  
 }  
  
 //ΜΕΝΟΥ ΓΙΑ ΧΡΗΣΤΗ BUYER  
 private void BrowseStore(Buyer customer, EShop eShop)  
 {  
 System.*out*.println("\t\t\tE-Shop\t\t\t");  
  
 eShop.showCategories();  
 System.*out*.println("\n" + "------------------------------------");  
  
 String continueBrowse = "y";  
 while(continueBrowse.equalsIgnoreCase("y"))  
 {  
 System.*out*.println("Which category of products you would like to see(full name)?");  
 String ans = keyboard.nextLine();  
 String copyAns = ans;  
  
 String keepCategory = "y";  
 while (keepCategory.equalsIgnoreCase("y"))  
 {  
 String catOK = eShop.showProductsInCategory(ans);  
 while(!(catOK.equalsIgnoreCase("ok")))  
 {  
 System.*out*.println("No such category in our list! \nPlease re-enter the category you want to see.");  
 ans = keyboard.nextLine();  
 catOK = eShop.showProductsInCategory(ans);  
 copyAns = ans;  
 }  
 String item = eShop.showProduct();  
 if(item.equalsIgnoreCase("empty\_category"))  
 {  
 System.*out*.println("This category is empty.");  
 System.*out*.println("Would you like to remain in the same category?(y/n)");  
 keepCategory = keyboard.nextLine();  
 if (keepCategory.equalsIgnoreCase("y")) {  
 keepCategory = "y";  
 ans = copyAns;  
 } else  
 keepCategory = "n";  
 }  
 else {  
 while (item.equalsIgnoreCase("not in list")) {  
 System.*out*.println("Enter a product from the list!");  
 item = eShop.showProduct();  
 }  
  
 System.*out*.println("Would you like to add that product in your cart? (y/n)");  
 String answer = keyboard.nextLine();  
 if (answer.equalsIgnoreCase("y")) {  
 customer.placeOrder(eShop, item);  
 }  
  
 System.*out*.println("Would you like to remain in the same category?(y/n)");  
 keepCategory = keyboard.nextLine();  
 if (keepCategory.equalsIgnoreCase("y")) {  
 keepCategory = "y";  
 ans = copyAns;  
 } else  
 keepCategory = "n";  
 }  
 }  
  
 System.*out*.println("Would you like to continue browsing at the store?(y/n)");  
 continueBrowse = keyboard.nextLine();  
 }  
  
 }  
  
 private boolean ViewCart(EShop eShop,Buyer buyer)  
 {  
 System.*out*.println("----------------------------------------" +  
 "\n" + "Your cart: ");  
 String cart\_state = buyer.getBuyerCart();  
 if(cart\_state.equalsIgnoreCase("empty"))  
 {  
 System.*out*.println("Add items in your cart.");  
 return false;  
 }  
 else {  
 System.*out*.println("---------------------------------");  
 System.*out*.println("How would you like to proceed?");  
 System.*out*.println("1. Change an item's quantity or remove an item from your list" + "\n2. Clear your cart" + "\n3. Check out");  
 System.*out*.print("Enter your option number(1-3): ");  
 int optNum = keyboard.nextInt();  
 String nLine = keyboard.nextLine();  
  
 while (optNum < 1 || optNum > 3) {  
 System.*out*.println("Option number must be between 1-3.");  
 System.*out*.print("Enter your option number(1-3): ");  
 optNum = keyboard.nextInt();  
 }  
  
 switch (optNum) {  
 case (1): {  
 String alterItemOrdered = "y";  
  
 while (alterItemOrdered.equalsIgnoreCase("y")) {  
 System.*out*.print("You want to remove or change the quantity of an item you ordered?(remove/change)");  
 String answer = keyboard.nextLine();  
 if (answer.equalsIgnoreCase("remove")) {  
 System.*out*.print("Which item would you like to remove from your list?");  
 String ans = keyboard.nextLine();  
 buyer.removeOrder(eShop, ans);  
 } else if (answer.equalsIgnoreCase("change")) {  
 System.*out*.println("Which item's quantity you would like to change?");  
 String ans = keyboard.nextLine();  
 System.*out*.print("You want to raise or lessen the quantity of that items?(r/l)");  
 String up\_low = keyboard.nextLine();  
 if (up\_low.equalsIgnoreCase("r")) {  
 System.*out*.print("Enter how many units you want to add: ");  
 int units = keyboard.nextInt();  
 String nChar = keyboard.nextLine();  
 buyer.changeQuantity(eShop, ans, units);  
 } else if(up\_low.equalsIgnoreCase("l")){  
 System.*out*.print("Enter how many units you want to subtract: ");  
 int units = keyboard.nextInt();  
 String nChar = keyboard.nextLine();  
 buyer.changeQuantity(eShop, ans, -units);  
 }  
  
 }  
  
 System.*out*.print("Would you like to make another change or remove an item from your cart?(y/n)");  
 alterItemOrdered = keyboard.nextLine();  
 }  
 return false;  
  
 }  
  
 case (2): {  
 buyer.clearBuyersCart(eShop);  
 return false;  
  
 }  
  
 case (3): {  
 boolean done = buyer.checkOutBuyer();  
 if(done)  
 {  
 return true;  
 }  
 else { return false;}  
 }  
 }  
 return false;  
 }  
 }  
  
 private boolean CheckOut(Buyer buyer)  
 {  
 return buyer.checkOutBuyer();  
 }  
  
 private int Back(int optionNum)  
 {  
 return optionNum-1;  
 }  
  
 private void LogOut()  
 {  
 System.*out*.println("Connect with another account?(y/n)");  
 String answer = keyboard.nextLine();  
 if(answer.equalsIgnoreCase("n"))  
 {  
 Exit();  
 }  
 }  
  
 private void Exit()  
 {  
 System.*out*.println("Thank you for visiting our e-shop." + '\n' + "We hope to visit us again.");  
 System.*exit*(0);  
 }  
  
  
 //MENU ΓΙΑ ΧΡΗΣΤΗ OWNER  
  
 private void ownerOptions(Owner owner,EShop shop)  
 {  
 System.*out*.println("Hello " + owner.getUserName() + " !");  
 System.*out*.println("Name: " + owner.getUserName() + ", Email: " + owner.getUsersEmail() + ", Administrator: " + owner.getUserDetails());  
 System.*out*.println("----------------------------------");  
 System.*out*.println("What would you like to do?");  
 System.*out*.println("1. Browse store\n" + "2. Check Status\n" + "3. Back\n" + "4. Log out\n" + "5. Exit");  
   
 System.*out*.println("----------------------------------");  
 System.*out*.println("With which option you would like to proceed (1-5)?");  
 int option = keyboard.nextInt();  
  
  
 boolean managing = true;  
 while(managing)  
 {  
  
 while(option<1 ||option>5)  
 {  
 System.*out*.print("Option number must be between 1 and 5!");  
 option = keyboard.nextInt();  
 }  
  
 switch(option)  
 {  
 case(1):  
 {  
 ownerBroseStore(owner,shop);  
 System.*out*.print("Would you like to go back/exit? (back/exit/no)" +  
 "\n(Warning! : choosing back from this point will return you to log-in screen.) ");  
 String back\_exit = keyboard.nextLine();  
 if(back\_exit.equalsIgnoreCase("back"))  
 {  
 LogOut();  
 managing = false;  
 }  
 else if(back\_exit.equalsIgnoreCase("exit"))  
 {  
 Exit();  
 }  
 else  
 {  
 System.*out*.print("Enter the option's number you want to continue with: ");  
 option = keyboard.nextInt();  
 break;  
 }  
 break;  
 }  
 case(2):  
 {  
 CheckStatus(shop);  
 System.*out*.print("Would you like to go back/exit? (back/exit/no)");  
 String back\_exit = keyboard.nextLine();  
 if(back\_exit.equalsIgnoreCase("back"))  
 {  
 option = Back(option);  
 break;  
 }  
 else if(back\_exit.equalsIgnoreCase("exit"))  
 {  
 Exit();  
 }  
 else  
 {  
 System.*out*.print("Enter the option's number you want to continue with: ");  
 option = keyboard.nextInt();  
 String line = keyboard.nextLine();  
 break;  
 }  
 break;  
 }  
 case(3):  
 {  
 option = Back(option);  
 break;  
 }  
 case(4):  
 {  
 managing = false;  
 String nullCHAR = keyboard.nextLine();  
 LogOut();  
 break;  
 }  
 case(5):  
 {  
 Exit();  
 break;  
 }  
 }  
 }  
  
  
 }  
  
 private void ownerBroseStore(Owner owner, EShop shop)  
 {  
 String nCHAR = keyboard.nextLine();  
 System.*out*.println("\t\t\tE-Shop\t\t\t");  
  
 shop.showCategories();  
 System.*out*.println("\n" + "------------------------------------");  
  
 String continueBrowse = "y";  
 while(continueBrowse.equalsIgnoreCase("y"))  
 {  
  
 System.*out*.print("Which category of products you would like to see?");  
 String ans = keyboard.nextLine();  
 String copyAns = ans;  
  
 String keepCategory = "y";  
 while (keepCategory.equalsIgnoreCase("y"))  
 {  
 String catOK = shop.showProductsInCategory(ans);  
 while(!(catOK.equalsIgnoreCase("ok")))  
 {  
 System.*out*.println("No such category in our list! \nPlease re-enter the category you want to see.");  
 ans = keyboard.nextLine();  
 catOK = shop.showProductsInCategory(ans);  
 copyAns = ans;  
 }  
 String item = shop.showProduct();  
 if(item.equalsIgnoreCase("empty\_category"))  
 {  
 System.*out*.println("Tha category of products is empty.");  
 }  
 else {  
 while (item.equalsIgnoreCase("not in list")) {  
 System.*out*.println("Enter a product from the list!");  
 item = shop.showProduct();  
 }  
  
 System.*out*.println("Would you like to alter current item's stock?(y/n)");  
 String answer = keyboard.nextLine();  
 if (answer.equalsIgnoreCase("y")) {  
 shop.updateItemStock(item);  
 }  
  
 System.*out*.println("Would you like to remain in the same category?(y/n)");  
 keepCategory = keyboard.nextLine();  
 if (keepCategory.equalsIgnoreCase("y")) {  
 keepCategory = "y";  
 ans = copyAns;  
 } else  
 keepCategory = "n";  
 }  
 }  
  
 System.*out*.println("Would you like to continue browsing at the store?(y/n)");  
 continueBrowse = keyboard.nextLine();  
 }  
  
 }  
  
 private void CheckStatus(EShop shop)  
 {  
 int numberOfCustomers = shop.getNumberOfCustomers();  
 if(numberOfCustomers == 0)  
 {  
 System.*out*.println("There are no customers in list.");  
 String nCHAR = keyboard.nextLine();  
 }  
 else {  
 shop.checkStatus();  
  
 String nCHAR = keyboard.nextLine();  
 System.*out*.print("Would you like to check a buyer's cart?(y/n)");  
 String answer = keyboard.nextLine();  
  
 while (answer.equalsIgnoreCase("y")) {  
 System.*out*.print("Enter the email of the buyer: ");  
 String buyersEmail = keyboard.nextLine();  
 if (shop.CheckEmail(buyersEmail)) {  
 ((Buyer) shop.getUser(buyersEmail)).getBuyerCart();  
 System.*out*.println("----------------------------------------");  
 System.*out*.println("Would you like to remove that buyer from buyers list?(y/n)");  
 String remove = keyboard.nextLine();  
 if (remove.equalsIgnoreCase("y")) {  
 ((Buyer) shop.getUser(buyersEmail)).clearBuyersCart(shop);  
 System.*out*.println(shop.removeBuyer((shop.getUser(buyersEmail)).getUserName()));  
 }  
 }  
  
 System.*out*.println("Would you like to check another buyer's cart?(y/n)");  
 answer = keyboard.nextLine();  
 }  
 }  
 }  
}

**CLASS MAIN**

import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
  
 Scanner keyboard = new Scanner(System.*in*);  
  
 //ΔΗΜΙΟΥΡΓΟΥΜΕ ΤΗ ΛΙΣΤΑ ΤΩΝ ΠΡΟΪΟΝΤΩΝ (3 items per class)  
 EShop shop = new EShop();  
  
 boolean more = true;  
  
 while(more)  
 {  
 shop.addItem();  
 System.*out*.print("Add another item in shop list?(y/n)");  
 String answer = keyboard.nextLine();  
 if(answer.equalsIgnoreCase("n"))  
 more = false;  
 }  
  
 /\*Items to add in E-shop list  
 Pens: Name: Blue Pen , Price: 1.2 €, Stock: 150 , Color: blue , Tip Size: 0.03  
 Name: Red Pen , Price: 1.2 €, Stock: 150 , Color: red , Tip Size: 0.03  
 Name: Black Pen, Price: 1.2 €, Stock: 150 , Color: black , Tip Size: 0.03  
 Pencils: Name: PencilVane, Price: 2 €, Stock: 80, Tip Size: 0.05 , Type: HB  
 Name: Regular Pencil, Price: 1 €, Stock: 200, Tip Size: 0.05, Type: B  
 Name: Mechanical Pencil, Price: 1.7 €, Stock: 130, Tip Size: 0.02, Type: H  
 NoteBooks: Name: My notebook, Price: 4.5 €, Stock: 75, Number Of Sections: 4  
 Name: Green notebook, Price: 3 €, Stock: 125, Number of Sections: 3  
 Name: Blue notebook, Price: 2.5 €, Stock: 90, Number of Sections: 2  
 Paper: Name: White paper, Price: 0.1 €, Stock: 500, Weigh: 2, Pages: 500  
 Name: Blue paper, Price: 0.2 €, Stock: 350, Weigh: 2, Pages: 350  
 Name: Green paper, Price: 0.2 €, Stock: 400, Weigh: 2, Pages: 400  
 \*/  
  
  
  
  
 //ΔΗΜΙΟΥΡΓΟΥΜΕ 1 OWNER ΚΑΙ 2 BUYERS  
  
 more = true;  
 while(more)  
 {  
 shop.addBuyer();  
 System.*out*.println("Add another buyer on list?(y/n)");  
 String answer = keyboard.nextLine();  
 if(answer.equalsIgnoreCase("n"))  
 more = false;  
 }  
  
 /\*  
 Owner owner = new Owner("KOSTAS");  
 owner.setUserEmail("Owner\_Kostas@eshopmail.gr");  
  
 //BUYERS  
 Buyer buyer1 = new Buyer("JILL");  
 buyer1.setUserEmail("Its\_Jill@eshopmail.gr");  
  
 Buyer buyer2 = new Buyer("BILL");  
 buyer2.setUserEmail("Its\_Bill@eshopmail.gr");  
  
 \*/  
  
 Menu menu = new Menu();  
  
 menu.LogIn(shop);  
 }  
}