Report to the Amazon simulation project

Marayam Soleimani Shaldehi

Spring 2023

In this project I have created 11 classes (including Main class) which I'm going to explain individually.

Class Admin

Here we have the related attributes and encapsulation methods (setters and getters)

In this class first we import java.util.UUID to use this feature in making usernames and passwords.

In the setUsername and setPassword methods first I created an object from the UUID class and called randomUUID so what happens here is that whenever these setter methods are called a random uuid is made by this object and is set equal to the usernames and passwords. So in fact the user does not set a username or a password for her/himself but the system will.

What is the point of this action?

We know that usernames and passwords must be unique so if we want to let the user choose them for her/himself for sure there is a possibility that that thing is already used by another user so we need to check it if it is already used or not and if the answer is yes we have to tell the new user to come up with a new one but by using uuid we are highly sure that the usernames and passwords that are devoted by the system are unique and there is no need to check this fact.

In the setEmail method first I created a valid email pattern so the email that the user enters must match the pattern. Here I defined a Boolean variable called flag to check if these two matches or not so I put it equal to "email.matches(emailPattern)" and then I set a while (true) . if flag is true it breaks from the loop and " this.email = email; " and else it continues in the loop and gets the email as an input again and again until the user enters a valid email address and then breaks.

Method Admin: first gets an email then chooses username and password.

Method toString: returns the attributes to string (obvious)

Method defaultAdmin: it only contains a default username and password for the admins.

Class Seller

the important concepts of this class are already explained in the Admin class.

Class User

This class is a child for Admin class (it has username , password and email attributes and related methods)

So all the attributes and encapsulation stuff are obvious and for the phone number attribute the same algorithm of email is used.

Whenever I want to get or show the attributes that are for the parent class I can use keyword “super” and then call the related method from the super class (parent class)

Class Product

There are 10 categories for products so I have individualized two of them from the others (laptop and phone) and get inputs for these two individually in the Laptop and SmartPhone methods and for the other 8 categories all in the Product method.

As well I have the toString methods to convert these pieces of information to string and show in the output.

Method addQuantityByOne:

This will add the available quantity (sellerQuantity) by one.

It is used when a product is added that is exactly the same as a product that already exists so it will add its quantity by one.

Class Panels

This class is created to store the public static variables and arraylists that must be passed and used in multiple classes.

Class Menus

In this class the methods are only some menus that are mainly used in the main class.

Class UserOperations

CreateAccount:

An obj from User class created and User method from this class is called. And there is an int variable called flag . when it gets information that are in User class it goes through a loop and check if the phone number already exists in the other users in the users arraylist ,the flag gets increased by one and the user must change phone number (each account one number << one number can not be devoted to more than one account>>)

When the flag remains zero means that the number is unique and the user obj is added to theUsers arrlist.

login:

gets a username and a password as inputs and goes to a loop in theUsers and checks if a user exist with both of them .if yes login successful and the Boolean variable uloginChecker = true and else it is false

(this will be used in the Main class later to access the menus after the login)

(the concept of flag here is used the same as previous method and in all the methods from now on is the same so I refuse to keep explaining them in all the methods)

editPersonalInfo:

here gets a username or a password , goes to theUsers arrlist by a loop and checks if there is a user that has both items at the same time . then it will save that user's username to a variable called "userNameForUpdating" then it shows a menu for the items that can be changed and for each of them that is chosen it goes to theUsers arrlist using loop and if "userNameForUpdating" matches any of their usernames then its chosen item (address/email/phone number) will be replaced by the new form of that item that was earlier gotten as input:

i.setEmail(newEmail); :

changes the previous email to the new one

Panels.theUsers.set(Panels.theUsers.indexOf(i),i); :

Updates the changes that were operated on i in the arrlist

addToCart_LeaveComment:

first it shows a menu to choose whether the product that the user wants to add to cart or leave a comment is electronics or any other categories. (this menu is created in a method in sellersoperations class so first we need an obj from this class (SOobj))

if the choice is electronics it again consists of laptop and phone and the related menu the same as before is shown.

If laptop is chosen: it goes through a loop in theProducts arrlist and what I want is that it only shows the laptops (the toStringLaptop method from Product class) there is a fact here that in spite of individualizing laptops , phones and the rest of the products all of these are from Product class and all of them have the attributes of Product class but obviously we haven't set all the attributes for all of them.

For example the scent attribute or formula for example are null for laptops , phones or some categories.(there are more examples like this)

So when I refer to the items that are saved in theProduct arrlist , I'm referring to all of them (there are laptops, phones and other categories here but I only want to print out laptops so I must look for some features that the laptops have but others don't) so:

```
if (Objects.isNull(i.getCategory()) && Objects.isNull(i.getCameraQuality())) {
                    System.out.println(Panels.theProducts.indexOf(i) + i.toStringLaptop());
```

For laptops category and camera quality attributes are null so if i had both of these features it is laptop and can be printed in this section and in the next line it prints the index of i and i.toStringLaptop().

The same rule goes for the other two.

addToCart_LeaveCommentChoice:

too obvious to be explained

add:

in the addToCart_LeaveComment method the products are printed with their indices at the beginning of them. So here I ask the user to only enter that index and add it to cart (shoppingCart arrlist) in that way.

comment:

the same as add method we have indices and by entering that a comment will be added to the comments arrlist of that product with that index (comments arrlist is an attribute for all the Products)

searchByBrand:

it gets a brand as input goes to theProduct arrlist and prints out the items with that brand and then the

addToCart_LeaveCommentChoice to add any of them to cart or leaving a comment.

seeShoppingCart:

this method is explained in addToCart_LeaveComments method.

removeFromCart:

the same as add with the difference that it will remove from shopping cart:

shoppingCart.remove(i);

addQuantity:

in this method the number of an item that user wants from a product is gotten from the input and set that number and minus it from the available quantity.

finalPrice:

this will calculate the total price of each shopping cart.

It adds the (i.getPrice()*i.getUserQuantity()) for every products that are inside the shoppingcart together.

finalizeOrderWallet:

this will fix the wallets and shop total profit. (it is obvious)

fixProductQuantity:

here the quantities will be updated for the next user:

```
j.setSellerQuantity(j.getSellerQuantity() - j.getUserQuantity());
                j.setUserQuantity(0);
```

finalizeOrderList:

here all the items from shopping cart will go to orderslist (for each user) and the wholeorderlist(orderlists of all users):

Panels.theWholeOrders.add(j.toString() + " " + "Local date:" + localDate.toString() + " "+ "Total Price:" + getTotalPrice() + "Username:"+ i.getUserName());

And then the fixproductquantity is called here and the shopping cart will be cleared:

shoppingCart.clear();

seeFinalOrderList:

obvious (the similar methods already discussed)

seeWholeOrdersList:

obvious

OrderRequest:

Here I ask the user to enter password. His/her password will go in the orderREQUEST arrlist. And then the admins can see them and confirm.

authorizedOrders:

when admin confirms those password go to the .AcceptedOrderListsRequests arrlist and if the password is there it means that it is confirmed and then Boolean OrderConfirmChecker = true; (it is used in Main)

fundRequest:

the same as orderrequest

acceptedFunds:

the same as authorizedOrders

showUsers:

obvious

showProducts:

obvious

Class sellersOperations

CreateAccount:

explained

login:

explained

AuthorizingRequest:

the same as orderrequest

AuthorizedSellers:

the same as authorizedOrders

ProductMenu:

obvious

electronicsMenu:

obvious

showSellers:

obvious

sellerSeeWallet:

here the seller enters password . it goes to the sellers arrlist with a loop and shows the wallet of the seller with that password.

addProduct:

here products are added by the seller

because it was hard to explain here please check out the comments in the code body.

Class AdminOperations

addAdmin:

obvious

login:

explained

showSellerAuthorizingRequests:

here it will show the name of the sellers that are in the theSellersAuthorizingRequests arrlist.and in a while true the admin keeps entering the sellers that he wants to confirm and when finished he Presses 'o'. all those names are stored in the AcceptedSellerAuthorizingRequests arrlist and at the same time remove from the theSellersAuthorizingRequests arrlist. And when 'o' is pressed theSellersAuthorizingRequests arrlist will be cleared .

showFinalOrderingRequests:

the same as showSellerAuthorizingRequests..

showUsersFundingRequests:

the same as showSellerAuthorizingRequests but with this difference that an amount of fund is devoted.

UserSeeWallet:

The same as sellerseewallet

showAdmins:

obvious

Class Main

Explaining this class here is hard

Please check out the comments in the program