به نام خدا

گزارش پروژه نهایی درس برنامه سازی پیشرفته

دکتر وحیده مقتدایی

احسان پدرام ۹۹۲۳۶۰۳۴

نیمسال دوم تحصیلی ۱۴۰۲-۱۴۰۳

"ساخت یک فروشگاه آنلاین"

شرح پروژه:

```
import java.util.*;
```

از این کد برای ایمپوت کردن تمام کتابخانه‌های java.util استفاده می‌کنیم.

کلاس Product:

```
class Product {
    private final String name;
    private final double price;
    private final int inventory;
    private final String seller;
    private final String preview;
```

کلاس Product که کلاسی مخصوص محصولات می‌باشد را تشکیل داده و برای کپسوله‌سازی آن پارامترهای مورد نیاز که شامل نام، قیمت، موجودی، فروشنده محصولات و نمایه‌ای از محصول می‌باشد را به صورت پرایوت تعریف می‌کنیم.

```
public Product(String name, double price, int inventory, String seller) {
    this.name = name;
    this.price = price;
    this.inventory = inventory;
    this.seller = seller;
    this.preview = "";
}
```

در اینجا کانستراکتور کلاس product را نوشتیم و به دلیل تشابه اسمی پارامترهای تعریفی در بالا و پارامترهای کانستراکتور از this استفاده کردیم، یعنی می‌توانستیم با استفاده از نام متفاوت از نوشتن this صرف نظر کنیم که در این‌صورت فرق نمی‌کرد. به عنوان مثال در کانستراکتور به جای name از Name استفاده می‌کردیم.

```
    public  String getProductName() {
        return name;
    }
    public double getPrice() {
        return price;
    }
    public int getInventory() {
        return inventory;
    }
    public String getSeller() {
        return seller;
    }
    public String getPreview() {
        return preview;
    }
}
```

حالا شروع به نوشتن getter های کلاس product می‌کنیم و برای هر پارامتر که در بالا نوشتیم با استفاده از تابع get آن را return می‌کنیم یا به قولی تابعی می‌سازیم که مقدار پارامتر را برگردانیم.

کلاس Order:

```
class Order {
    private final int orderId;
    private final Account user;
    private final List<Product> products;
    private double totalPrice;
```

کلاس Order که کلاسی مخصوص سفارش‌ها شامل پارامترهای آیدی، حساب کاربری، لیست محصولات و مجموع قیمت سفارش می‌باشد را تشکیل داده و برای کپسوله کردن، آن‌ها را پرایوت تعریف می‌کنیم.

```
public Order(int orderId, Account user, List<Product> products) {
    this.orderId = orderId;
    this.user = user;
    this.products = products;
    this.totalPrice = 0.0;
}
```

در اینجا کانستراکتور کلاس Order را نوشتیم و مقدار صفر را به مجموع قیمت لیست سفارش اختصاص دادیم تا بعدا با محاسبه قیمت داخل این پارامتر بریزیم. **می‌دانیم که نام سازنده باید با نام کلاس یکی باشد.**

```
    public int getOrderId() {
        return orderId;
    }
    public Account getUser() {
        return user;
    }
    public List<Product> getProducts() {
        return products;
    }
    public double getTotalPrice() {
        for (Product product : products) {
            totalPrice += product.getPrice();
        }
        return totalPrice;
    }
}
```

مثل کلاس قبل اینبار هم شروع به نوشتن getter های کلاس Order می‌کنیم و برای هر پارامتر که در بالا نوشتیم با استفاده از تابع get آن را برمی‌گردانیم؛ همچنین به کمک حلقه for و کلاس product که می‌توان قیمت هر محصول را از آن استخراج کرد متدی برای به دست آوردن قیمت لیست محصولاتمان نوشتیم.

کلاس Wallet:

```
class Wallet {
    private double inventory;

    public Wallet() {
        this.inventory = 0.0;
    }
    public double getInventory() {
        return inventory;
    }
    public void increaseInventory(double amount) {
        inventory += amount;
```

```
    }
    public void decreaseInventory(double amount) {
        inventory -= amount;
    }
}
```

کلاس Wallet یا کیف پول را که کلاسی مخصوص دخل و خرج است می‌نویسیم؛ این کلاس شامل موجودی حساب است که مقدار آن را در ابتدا صفر در نظر گرفتیم و در ادامه سه تابع برای برگرداندن مقدار موجودی کیف پول وهمچنین افزایش و کاهش مقدار آن نوشتیم.

کلاس Recommend:

```
class Recommend {
    private final List<Order> orders;
    private final List<Wallet> wallets;
    Recommend() {
        this.orders = new ArrayList<>();
        this.wallets = new ArrayList<>();
    }
    public void addOrder(Order order) {
        orders.add(order);
    }
    public void addWallet(Wallet wallet) {
        wallets.add(wallet);
    }
    public  Wallet getWallet(int index) {
        return wallets.get(index);
    }
    public  Order getOrder(int index) {
        return orders.get(index);
    }
}
```

کلاس Recommend کلاسی مخصوص درخواست‌هاست که شامل دو بخش سفارش‌ها و اعتبار کیف پول‌ها می‌باشد؛ در ادامه خواهیم دید که با تعریف سه نوع کاربر شامل مدیر، فروشنده و خریدار به واسطه این کلاس رابطه بین این انواع کارها برقرار می‌شود، به طوری که فروشنده و خریدار جهت اضافه و یا کم کردن محصولات و یا کاهش و افزایش مقدار موجودی کیف پول باید به ادمین درخواست دهند تا او تایید و یا رد کند.

کلاس Account:

```
abstract class Account {
    private String UserName;
    private String UserPass;
    public enum UserPermission { admin, seller, customer }
    private UserPermission Permission;
    private String UserMail;
    private String UserPhone;
    private String UserAddress;
    private final List<Order> OrderList;
    private final Wallet UserWallet;
    private String CompanyName;
    private final List<Recommend> recommends;
    public Account(String userName, String userPass, UserPermission
```

```
userPermission, String userMail) {
        this.UserName = userName;
        this.UserPass = userPass;
        this.Permission = userPermission;
        this.UserMail = userMail;
        recommends = new ArrayList<>();
        OrderList = new ArrayList<>();
        UserWallet = getWallet();
    }
```

کلاس انتزاعی Account را با پارامترهای قابل مشاهده در کد بالا مانند کلاس‌های قبلی تعریف می‌کنیم و با استفاده از دستور enum سه سطح دسترسی مختلف با نام‌های مدیر، فروشنده و خریدار ایجاد می‌کنیم.

```
public static void logIn(Shop shop) {
    int indexOfUser;
    String userName, userPassword;
    Scanner input = new Scanner(System.in);
    ArrayList<String> AccountNameList = new ArrayList<>();
    for (Account user : shop.getUsers())
AccountNameList.add(user.getUserName());
    while (true) {
        System.out.println("Please enter username:(Enter 0 to back)");
        userName = input.nextLine();
        if(userName.equals("0"))
            return;

        System.out.println("Please enter password:");
        userPassword = input.nextLine();

        indexOfUser = AccountNameList.indexOf(userName);
        if(indexOfUser != -1) {
            if
(shop.getUsers().get(indexOfUser).getUserPass().equals(userPassword)) {
                switch (shop.getUsers().get(indexOfUser).getUserPermission())
{
                    case admin:
                        shop.adminMenu();
                        break;
                    case seller:
                        shop.sellerMenu(shop.getUsers().get(indexOfUser));
                        break;
                    case customer:
                        shop.customerMenu(shop.getUsers().get(indexOfUser));
                        break;
                }
            } else
                System.out.println("password is not correct! :)");
        } else
            System.out.println("username is not exist! :)");
    }
}
```

در این کلاس نیاز به تابعی برای ورود کاربران نیاز داریم، متد بالا یک تابع با تمام جزئیات برای ورود کاربر به فروشگاه ما می‌باشد. از چالش‌های اساسی که در این نقطه با آن مواجه شدم نحوه تکرار این فرایند و همچنین فرایندهای بعدی که نیاز به تکرار مکرر در صفحه فروشگاه بود که با استفاده از حلقه همواره درست این مشکل را رفع کردم. در اینجا نام کاربری و رمز عبور و همچنین احراز عدم ثبت‌نام تکراری انجام شده است و به واسطه دستور سوئیچ کیس، سه منو برای سه نوع کاربری که اشاره کردیم ایجاد کردیم که این منو ها در کلاس Shop به زودی معرفی خواهند شد.

```
public static void signIn(Shop shop) {
    int indexOfUser;
    String userName, userPassword, accountType, userEmail, userPhone,
UserAddress;
    Scanner input = new Scanner(System.in);
    ArrayList<String> AccountNameList = new ArrayList<>();
    for (Account user : shop.getUsers())
AccountNameList.add(user.getUserName());
    while (true) {
        System.out.println("Please enter username:(Enter 0 to back)");
        userName = input.nextLine();
        if(userName.equals("0")  || userName.isEmpty())
            return;

        System.out.println("Please enter password:(Enter 0 to back)");
        userPassword = input.nextLine();
        if(userPassword.equals("0")  || userPassword.isEmpty())
            return;

        indexOfUser = AccountNameList.indexOf(userName);
        if(indexOfUser == -1) {
            System.out.println("1. customer or 2. seller?(Enter 1 or 2 or
enter 0 to back)");
            accountType = input.nextLine();
            if(accountType.equals("0") || accountType.isEmpty())
                return;
            switch(accountType) {
                case "1":
                    System.out.println("Please Enter Email address?(Enter 0
to back)");
                    userEmail = input.nextLine();
                    if(userEmail.equals("0") || userEmail.isEmpty())
                        return;
                    System.out.println("Please Enter phone number?(Enter 0 to
back)");
                    userPhone = input.nextLine();
                    if(userPhone.equals("0") || userPhone.isEmpty())
                        return;
                    System.out.println("Please Enter address?(Enter 0 to
back)");
                    UserAddress = input.nextLine();
                    if(UserAddress.equals("0") || UserAddress.isEmpty())
                        return;
                    Customer customer = new Customer(userName, userPassword,
UserPermission.customer,
                            userEmail, userPhone, UserAddress);
                    shop.addUser(customer);
                    System.out.println("your Account successfully created");
                    return;
                case "2":
                    String companyName;
                    System.out.println("Please Enter Email address?(Enter 0
to back)");
                    userEmail = input.nextLine();
                    if(userEmail.equals("0") || userEmail.isEmpty())
                        return;
                    System.out.println("Please Enter phone number?(Enter 0 to
back)");
```

```
                    userPhone = input.nextLine();
                    if(userPhone.equals("0") || userPhone.isEmpty())
                        return;
                    System.out.println("Please Enter address?(Enter 0 to
back)");
                    UserAddress = input.nextLine();
                    if(UserAddress.equals("0") || UserAddress.isEmpty())
                        return;
                    System.out.println("Please Enter company name?(Enter 0 to
back)");
                    companyName = input.nextLine();
                    if(companyName.equals("0") || companyName.isEmpty())
                        return;
                    Seller seller = new Seller(userName, userPassword,
UserPermission.seller,
                            userEmail, userPhone, UserAddress, companyName);
                    shop.addUser(seller);
                    System.out.println("Account has built.");
                    return;
            }
        } else
            System.out.println("This username has already been used! :)");
    }
}
```

متد signin هم مانند تابع مخصوص ورود کاربران نوشته شده است با این تفاوت که در اینجا اطلاعات کامل‌تری مانند ایمیل، شماره تماس، آدرس و یا نام شرکت برای فروشندگان از کابر دریافت می‌کنیم. دستورهای استفاده شده دقیقا مشابه ورود کاربران است و نیاز به توضیح ندارد.

```
   public String getUserName() {
        return UserName;
    }
    public String getUserPass() {
        return UserPass;
    }
    public UserPermission getUserPermission() {
        return Permission;
    }
    public String getUserMail() {
        return UserMail;
    }
    public String getUserPhone() {
        return UserPhone;
    }
    public String getUserAddress() {
        return UserAddress;
    }
    public List<Order> getOrderList() {
        return OrderList;
    }
    public Wallet getWallet() {
        return UserWallet;
    }
    public void setUserName(String userName) {
        UserName = userName;
    }
    public void setUserPass(String userPass) {
        UserPass = userPass;
```

```
    }
    public void setUserPermission(UserPermission userPermission) {
        Permission = userPermission;
    }
    public void setUserMail(String userMail) {
        UserMail = userMail;
    }
    public void setUserPhone(String userPhone) {
        UserPhone = userPhone;
    }
    public void setUserAddress(String userAddress) {
        UserAddress = userAddress;
    }
    public String getCompanyName() {
        return CompanyName;
    }
    public void setCompanyName(String companyName) {
        CompanyName = companyName;
    }
    public void addRecommend(Recommend recommend) {
        recommends.add(recommend);
    }
    public List<Recommend> getRecommends() {
        return recommends;
    }
    public void deleteRecommend(int index) {
        recommends.remove(index);
    }
}
```

در اینجا هم باز دقیقا مثل دیگر کلاس‌ها توابع get را برای پارامترهایمان نوشتیم.

زیر کلاس‌های Admin & seller & customer:

```
class Admin extends Account {
    public Admin(String username, String password, UserPermission permission,
String email) {
        super(username, password, permission, email);
    }
}
```

زیرکلاس Admin را که متعلق به مدیران فروشگاه آنلاین ما می‌باشد را زیرکلاسی از کلاس انتزاعی Account تعریف می‌کنیم و با دستور super اطلاعات کلاس والد را حفظ می‌کنیم.

```
class Seller extends Account {
    public Seller(String username, String password, UserPermission
permission, String email, String phone, String address, String companyName) {
        super(username, password, permission, email);
        setUserPhone(phone);
        setUserAddress(address);
        setCompanyName(companyName);
    }
}
```

زیرکلاس Seller را که متعلق به فروشندگان ما می‌باشد را با همان توضیحات زیرکلاس قبلی زیرکلاسی از کلاس انتزاعی Account معرفی می‌کنیم با این تفاوت که چند ویژگی مخصوص این نوع کاربر در زیر این کلاس اضافه می‌شوند.

```
class Customer extends Account {
    public Customer(String username, String password, UserPermission
permission, String email, String phone, String address) {
        super(username, password, permission, email);
        setUserPhone(phone);
        setUserAddress(address);
    }
}
```

زیرکلاس Customer را که متعلق به خریداران ما می‌باشد را با همان توضیحات زیرکلاس‌های قبلی زیرکلاسی از کلاس انتزاعی Account معرفی می‌کنیم با این تفاوت که چند ویژگی مخصوص این نوع کاربر در زیر این کلاس اضافه می‌شوند.

کلاس Shop:

```
class Shop {
    private String ShopName;
    private String WebAddress;
    private String SupportNumber;
    private final List<Account> Users;
    private final List<Product> Products;
    private final List<Order> Orders;
    private final Wallet shopWallet;

    public Shop(String shopName, String webAddress, String supportNumber) {
        setShopName(shopName);
        setWebAddress(webAddress);
        setSupportNumber(supportNumber);
        this.Users = new ArrayList<>();
        this.Products = new ArrayList<>();
        this.Orders = new ArrayList<>();
        this.shopWallet = new Wallet();
    }
```

کلاس Shop که بزرگترین کلاس ما هم به حساب می‌آید را هم مانند کلاس‌های قبلی کپسوله کردیم.

حال همانطور که گفته شد ما نیاز به سه منو برای سه نوع کاربر متفاوت خود داریم که با نام‌های adminmenu و sellermenu و customermenu آنها را معرفی می‌کنیم و برای هر کدام یک متد می‌نویسیم.

به دلیل زیاد نشدن صفحات گزارش توضیحات و چالش‌های این بخش از کد را بدون آوردن کد ارائه خواهم کرد.

منوها به صورت چند دستور تو در توی سوئیچ کیس نوشته شده‌اند که وابسته به سطح دسترسی و فعالیت‌های هر نوع کاربر این منوها پیاده شده‌اند.

به عنوان مثال منوی ادمین دارای فعالیت‌های زیر است:

```
System.out.println("admin menu:");
System.out.println("""
        1. Users List
        2. Products List
        3. Orders List
        4. Total Profit
        5. recommends List
        0. logOut""");
```

همچنین فعالیت‌های دو نوع کاربر دیگر به شکل زیر می‌باشند:

```
System.out.println("seller menu:");
System.out.println("""
        1. Products List
        2. Account view
        3. recommends List
        0. logOut""");

System.out.println("customer menu:");
System.out.println("""
        1. Products List
        2. Buy Basket
        3. recommends List
        4. Account view
        0. logOut""");
```

که هرکدام از این منوها خود شامل منوی دیگری می‌باشد.

یکی دیگر از چالش‌های جدی در این قسمت مواجه شدن با فرم‌های بسیار تو در تو به واسطه هر منویی که از طریق منویی دیگر به وجود می‌آمد بود که می‌بایست به واسطه نام‌گذری دقیق و رعایت نظم آن‌ها را به خوبی جلو برد.

کلاس OnlineStore:

```
class OnlineStore {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Shop shop = new Shop("Ehsan shop","https://ehsan-shop.ir","021-
45623478");
        Admin admin = new
Admin("admin","admin",Account.UserPermission.admin,"admin@gmail.com");
        shop.addUser(admin);
        while(true) {
            System.out.println("\t*************************\nWelcome to "
+ shop.getShopName() + "\nweb address: " +
                    shop.getWebAddress() + "\nSupport phone: " +
shop.getSupportNumber() + "\n\t*************************");
            System.out.println("Select from menu:\n1. login\n2. sign
in\n0. exit");
            switch (input.nextInt()) {
                case 0:
                    return;
                case 1:
                    Account.logIn(shop);
                    break;
                case 2:
                    Account.signIn(shop);
                    break;
                default:
                    break;
            }
        }
    }
}
```

کلاس OnlineStore که کلاس اصلی و نهایی ما می‌باشد را ابتدا با اطلاعات اولیه فروشگاه مانند نام و آدرس فروشگاه که خواسته سوال بود شروع کردیم و سپس باز با همان حلقه همواره درست و دستور سوئیچ کیس منوی اصلی را ساخته و ربای ثبت نام و یا ورود افراد را به کلاس Account ارجاع دادیم.

هر کاربر ابتدا باید ثبت‌نام کند و سپس دوباره وارد این لوپ شده و اصطلاحا login کند.

پایان