# گزارش پروژه پایانترم جاوا – آنلاین شاپ


**شایان وفائی**

**دانشجوی ریاضیات و کاربرد ها**

**شماره دانشجویی : ۹۹۲۲۲۱۱۵**


**طراحی کلاس آنلاین شاپ**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
class OnlineShop {

    // here we define the attributes
    private String name; نام فروشگاه
    private String webAddress; آدرس وب
    private String supportNumber; شماره پشتیبانی
    private List<Product> products; لیست محصولات
    private List<Order> orders; لیست سفارشات
    private double totalProfit;آخرین مجموع سود فروشگاه
    List<User> users; لیست حساب کاربران
```

حال سازنده این کلاس را معرفی میکنیم و مقادیر اولیه ای به المان ها نسبت می دهیم.

```
    // here we define the method of the Onlineshop constructor
    public OnlineShop(String name, String webAddress, String supportNumber) {
        this.name = name;
        this.webAddress = webAddress;
        this.supportNumber = supportNumber;
        this.users = new ArrayList<>();
        this.products = new ArrayList<>();
        this.orders = new ArrayList<>();
        this.totalProfit = 0;
    }
```

باید متودی برای ثبت نام کاربران تعریف کنیم.

```
    // here we define the method to register new users
    public void registerUser(User user) {
        users.add(user);
        System.out.println("User registered successfully.");
    }
```

پس از آن باید متودی برای وارد شدن به حساب کاربری معرفی کنیم.

```
    // here we define the method to login users
    public User loginUser(String username, String password) {
        for (User user : users) {
            if (user.getUsername().equals(username) &&
user.getPassword().equals(password)) {
                System.out.println("Login successful.");
                return user;
            }
        }
        System.out.println("Invalid username or password.");
        return null;
    }
```

باید متودی برای اضافه کردن محصولات معرفی کنیم.

```
// here we define the method to add products to the products list
public void addProduct(Product product) {
    products.add(product);
    System.out.println("Product added successfully.");
}
```

باید متودی برای گرفتن و مشاهده لیست محصولات معرفی کنیم.

```
// here we define the method to get products list
public void listProducts() {
    for (Product product : products) {
        System.out.println(product);
    }
}
```

باید متودی برای اضافه کردن سفارش مشتریان به لیست سفارشات معرفی کنیم.

```
// here we define the method to add orders of the customers
public void addOrder(Order order) {
    orders.add(order);
    System.out.println("Order placed successfully.");
}
```

باید متودی برای تایید سفارشات توسط ادمین فروشگاه معرفی کنیم.

```
// here we define the method to approve order
public void approveOrder(Order order) {
    if (order.isApproved()) {
        System.out.println("Order is already approved.");
    } else {
        order.setApproved(true);
        totalProfit += order.getTotalCost() * 0.1;
        System.out.println("Order approved successfully.");
    }
}
```

باید متودی برای مشاهده مجموع سود فروشگاه معرفی کنیم.

```
// here we define the method to view the total profit of the shop
 public void viewTotalProfit() {
     System.out.println("Total profit: $" + totalProfit);
 }
```

باید متودی برای ثبت نام ادمین جدید معرفی کنیم.

```
 // here we define the method to get register new admins
 public void registerAdmin(Admin admin) {
     users.add(admin);
     System.out.println("Admin registered successfully.");
 }
```

باید متودی برای مشاهده لیست محصولات موجود معرفی کنیم.

```
// here we define the method to get products list
 public void listProductsWithDetails() {
     for (Product product : products) {
         System.out.println(product.getDetailedInfo());
     }
 }
```

باید متودی برای دسترسی محصولات از لیست محصولات معرفی کنیم.

```
 // here we define the method to get products list
 public List<Product> getProducts() {
     return products;
 }
```

باید متودی برای دسترسی به کاربران از لیست کاربران معرفی کنیم.

```
 // here we define the method to get users list
 public List<User> getUsers() {
     return new ArrayList<>(users);
 }
```

**طراحی کلاس کاربران**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the class of Users to define the user objects
abstract class User {
    protected String username; نام کاربری
    protected String password; پسوورد
    protected String email; ایمیل کاربری
    protected String phoneNumber; شماره تلفن
    protected String address;آدرس کاربر
    protected ShoppingCart cart; سبد خرید کاربر
    protected List<Order> orders; سفارشات کابرد
    protected List<Product> purchasedProducts; أقلام خرید شده کاربر
    protected double wallet; کیف پول کاربر
```

متود سازنده کلاس کاربران را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method to construct the users attributes
    public User(String username, String password, String email, String phoneNumber,
String address) {
        this.username = username;
        this.password = password;
        this.email = email;
        this.phoneNumber = phoneNumber;
        this.address = address;
        this.cart = new ShoppingCart();
        this.orders = new ArrayList<>();
        this.purchasedProducts = new ArrayList<>();
        this.wallet = 0;
    }
```

متودی برای ثبت سفارش معرفی می کنیم.
ابتدا لیست سبد خرید کاربر را دریافت می کند، هزینه کل خرید را محاسبه می کند، با موجودی کالا بررسی می کند، موجودی کیف پول کاربر را بررسی می کند، و در صورت درست بودن همه موارد، ثبت سفارش را تایید می کند.

```
// here we define the method to place an order (list of customers shopping cart to be
purchased)
```

```
    public boolean placeOrder(OnlineShop shop) {
        if (cart.getProducts().isEmpty()) {
            System.out.println("Cart is empty. Cannot place an order.");
            return false;
        }

        Order order = new Order(this, cart.getProducts());
        if (wallet >= order.getTotalCost()) {

            // Check if there's enough inventory for all products
            for (Map.Entry<Product, Integer> entry : cart.getProducts().entrySet()) {
                Product product = entry.getKey();
                int quantity = entry.getValue();
                if (product.getInventory() < quantity) {
                    System.out.println("Not enough inventory for " +
product.getName());
                    return false;
                }
            }









            // Reduce inventory and add order
            for (Map.Entry<Product, Integer> entry : cart.getProducts().entrySet()) {
                Product product = entry.getKey();
                int quantity = entry.getValue();
                product.reduceInventory(quantity);
            }

            orders.add(order);
            shop.addOrder(order);
            wallet -= order.getTotalCost();
            cart.clearCart();
            return true;
        } else {
            System.out.println("Insufficient funds. Order total: $" +
order.getTotalCost() + ", Wallet balance: $" + wallet);
            return false;
        }
    }
```

متودی برای گرفتن نام کابری معرفی می کنیم.

```
// here we define the method to get user's username
public String getUsername() {
    return username;
}
```

متودی برای گرفتن پسوورد معرفی می کنیم.

```
// here we define the method to get user's password
public String getPassword() {
    return password;
}
```

متودی برای افزایش موجودی کیف پول معرفی می کنیم.

```
// here we define the method to add funds to user's balance
public void addFunds(double amount) {
    wallet += amount;
    System.out.println("Funds added successfully. New balance: $" + wallet);
}
```

متودی برای مشاهده پروفایل کاربر معرفی می کنیم.

```
// here we define the method to get user's profile to view
public void viewProfile() {
    System.out.println("Username: " + username);
    System.out.println("Email: " + email);
    System.out.println("Phone Number: " + phoneNumber);
    System.out.println("Address: " + address);
    System.out.println("Wallet Balance: $" + wallet);
}
```

متودی برای اضافه کردن محصولات انتخابی کاربر به سبد خرید او معرفی می کنیم.

```
    // here we define the method to add user's wanted products to his/her cart
    public void addToCart(Product product, int quantity) {
        cart.addProduct(product, quantity);
        System.out.println("Product added to cart.");
    }
```

متودی برای مشاهده سبد خرید کاربر معرفی می کنیم.

```
    // here we define the method to get user's cart items
    public void viewCart() {
        cart.viewCart();
    }
```

متودی برای فرستادن درخواست افزایش موجودی کیف پول به ادمین جهت تایید معرفی می کنیم.

```
    // here we define the method to send fund requests to admins
    public void requestFunds(double amount, List<Admin> admins) {
        System.out.println("Fund request of $" + amount + " sent to administrators.");
        for (Admin admin : admins) {
            admin.receiveFundRequest(this, amount);
        }
    }
```

**طراحی زیر کلاس ادمین از کلاس کاربر**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the sub-class of Admins to define the admin objects
class Admin extends User {
    private List<FundRequest> fundRequests;
```

متود سازنده زیر کلاس ادمین را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the Admin constructor
    public Admin(String username, String password, String email, String phoneNumber,
String address) {
        super(username, password, email, phoneNumber, address);
        this.fundRequests = new ArrayList<>();
    }
```

متودی برای ثبت نام ادمین جدید معرفی می کنیم.

```
    // here we define the method to register new admins
    public void registerNewAdmin(OnlineShop shop, Admin newAdmin) {
        shop.registerAdmin(newAdmin);
    }
```

متودی برای دریافت درخواست افزایش موجودی کیف پول کابر را معرفی می کنیم.

```
    // here we define the method to receive fund requests
    public void receiveFundRequest(User user, double amount) {
        fundRequests.add(new FundRequest(user, amount));
    }
```

متودی برای مشاهده درخواست افزایش موجودی کیف پول کابر جهت تایید را معرفی می کنیم.

```
    // here we define the method to view fund requests to be approved by admin
    public void viewFundRequests() {
        for (int i = 0; i < fundRequests.size(); i++) {
            System.out.println((i + 1) + ". " + fundRequests.get(i));
        }
    }
```

متودی جهت تایید درخواست فوق معرفی میکنیم.

```
    // here we define the method to approve fund requests
    public void approveFundRequest(int index) {
        if (index >= 0 && index < fundRequests.size()) {
            FundRequest request = fundRequests.get(index);
            request.getUser().addFunds(request.getAmount());
            fundRequests.remove(index);
            System.out.println("Fund request approved and processed.");
        } else {
            System.out.println("Invalid request index.");
        }
    }
```

متودی برای تایید فروشنده توسط ادمین معرفی می کنیم.

```
    // here we define the method to approve seller
    // In the Admin class
    public void approveSeller(Seller seller) {
        seller.setApproved(true);
        System.out.println("Seller approved successfully.");
    }
```

**طراحی زیر کلاس فروشنده از کلاس کاربر**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the sub-class of sellers to define the seller objects
class Seller extends User {
    private String companyName;نام فروشنده
    private List<Product> available Products;لیست کالا های فروشنده
    private boolean isApproved; تاییدیه
```

متود سازنده زیر کلاس فروشنده را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the seller constructor
    public Seller(String companyName, String username, String password, String email,
String phoneNumber, String address) {
        super(username, password, email, phoneNumber, address);
        this.companyName = companyName;
        this.availableProducts = new ArrayList<>();
        this.isApproved = false;
    }
```

متودی جهت تایید معرفی می کنیم.

```
    // here we define the method to approve
    public void setApproved(boolean isApproved) {
        this.isApproved = isApproved;
    }
```

متودی برای اضافه کردن محصولات توسط فروشنده به لیست محصولات معرفی می کنیم.

```
    // here we define the method to add products to sellers list of products
    public void addProduct(Product product, OnlineShop shop) {
        if (isApproved) {
            availableProducts.add(product);
            shop.addProduct(product);
            System.out.println("Product added to shop successfully.");
            System.out.println("Added product details:");
            System.out.println(product.getDetailedInfo());
        } else {
            System.out.println("Seller is not approved to add products.");
        }
    }
```

**طراحی زیر کلاس مشتری از کلاس کاربر**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the sub-class of customers to define the customer objects
class Customer extends User {
```

متود سازنده زیر کلاس مشتری را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the customer constructor
    public Customer(String username, String password, String email, String
phoneNumber, String address) {
        super(username, password, email, phoneNumber, address);
    }
```

**طراحی کلاس محصولات**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the class of out products to create product objects
abstract class Product {
    protected String name;نـام مـحصول
    protected double price; هزیـنه مـحصول
    protected int inventory; مـوجودی مـحصول
    protected List<Review> reviews; نـظرات مـشتریـان
```

متود سازنده کلاس محصولات را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the product constructor
    public Product(String name, double price, int inventory) {
        this.name = name;
        this.price = price;
        this.inventory = inventory;
        this.reviews = new ArrayList<>();
    }
```

متودی برای کاهش موجودی محصولات بعد از خرید مشتری معرفی می کنیم.

```
    // here we define the method to add products to reduce inventory count after user
purchase
    public void reduceInventory(int quantity) {
        if (quantity <= this.inventory) {
            this.inventory -= quantity;
        } else {
            throw new IllegalArgumentException("Not enough inventory");
        }
    }
```

متودی برای گرفتن نام، هزینه محصولات و موجودی محصولات را معرفی می کنیم.

```
    // here we define the method to get sellers name
    public String getName() {
        return name;
    }

    // here we define the method to get products price
    public double getPrice() {
        return price;
    }

    // here we define the method to get products inventory count
    public int getInventory() {
        return inventory;
    }
```

متودی برای اضافه کردن نظرات مشتریان به محصولات را معرفی می کنیم.

```
    // here we define the method to add review to products
    public void addReview(Review review) {
        reviews.add(review);
        System.out.println("Review added successfully.");
    }
```

**طراحی زیر کلاس وسایل الکترونیکی**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the sub-class of electronics as a category of products
class Electronics extends Product {
```

متود سازنده کلاس محصولات را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the electronics constructor
    public Electronics(String name, double price, int inventory) {
        super(name, price, inventory);
    }
}
```

**طراحی زیر کلاس وسایل الکترونیکی از کلاس محصولات**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the sub-class of books as a category of products
class Books extends Product {
```

متود سازنده کلاس را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the books constructor
    public Books(String name, double price, int inventory) {
        super(name, price, inventory);
    }
```

**طراحی زیر کلاس لباس ها از کلاس محصولات**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the sub-class of clothings as a category of products
class Clothing extends Product {
```

متود سازنده کلاس را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the clothings constructor
    public Clothing(String name, double price, int inventory) {
        super(name, price, inventory);
    }
```

**طراحی زیر کلاس جواهرات از کلاس محصولات**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the sub-class of clothings as a category of products
class Clothing extends Product {
```

متود سازنده کلاس را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the jewelry constructor
    public Jewelry(String name, double price, int inventory) {
        super(name, price, inventory);
    }
```

**طراحی زیر کلاس خودرو ها از کلاس محصولات**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the sub-class of cars as a category of products
class Cars extends Product {
```

متود سازنده کلاس را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the cars constructor
    public Cars(String name, double price, int inventory) {
        super(name, price, inventory);
    }
```

**طراحی کلاس سبد خرید**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
    // here we define the method of the shopping cart constructor
    public ShoppingCart() {
        this.products = new HashMap<>();
    }
```

متود سازنده کلاس را معرفی کرده و مقدار اولیه می دهیم.

```
// here we define the class of shopping cart to have a hashmap of user intended
products to purchase
class ShoppingCart {
    private Map<Product, Integer> products;
```

متودی برای افزایش محصولات به سبد خرید اضافه می کنیم.

```
    // here we define the method to add products to shopping cart
    public void addProduct(Product product, int quantity) {
        if (product.getInventory() >= quantity) {
            products.put(product, products.getOrDefault(product, 0) + quantity);
            System.out.println("Product added to cart.");
        } else {
            System.out.println("Insufficient inventory for " + product.getName());
        }
    }
```

متودی برای مشاهده اقلام سبد خرید معرفی می کنیم.

```
    // here we define the method to view cart list
    public void viewCart() {
        if (products.isEmpty()) {
            System.out.println("Your cart is empty.");
        } else {
            for (Map.Entry<Product, Integer> entry : products.entrySet()) {
                System.out.println(entry.getKey().getName() + " - Quantity: " +
entry.getValue() + " - Total: $" + (entry.getKey().getPrice() * entry.getValue()));
            }
        }
    }
```

متودی برای گرفتن محصولات سبد خرید معرفی می کنیم.

```
    // here we define the method to get products
    public Map<Product, Integer> getProducts() {
        return new HashMap<>(products);
    }
```

متودی برای پاک کردن لیست سبد خرید معرفی می کنیم.

```
    // here we define the method to clear the shopping cart
    public void clearCart() {
        products.clear();
    }
```

**طراحی کلاس سفارش**

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the class of fund requests that sends the users
class Order {
    private User user;
    private Map<Product, Integer> products;
    private double totalCost;
    private boolean isApproved;
```

متود سازنده کلاس را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the order constructor
    public Order(User user, Map<Product, Integer> products) {
        this.user = user;
        this.products = new HashMap<>(products);
        this.totalCost = calculateTotalCost();
        this.isApproved = false;
    }
```

متودی برای محاسبه کل هزینه سبد خرید معرفی می کنیم.

```
    // here we define the method to calculate the total cost of the purchases
    private double calculateTotalCost() {
        return products.entrySet().stream()
            .mapToDouble(entry -> entry.getKey().getPrice() * entry.getValue())
            .sum();
    }
```

متودی برای گرفتن کل هزینه سبد خرید محاسبه می کنیم.

```
    // here we define the method to get total cost
    public double getTotalCost() {
        return totalCost;
    }
```

### طراحی کلاس نظرات

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the class of review to add reviews for the products
class Review {
    private String reviewer;
    private String comment;
    private int rating;
```

متود سازنده کلاس را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the review constructor
    public Review(String reviewer, String comment, int rating) {
        this.reviewer = reviewer;
        this.comment = comment;
        this.rating = rating;
    }
```

### طراحی کلاس درخواست فاند

در این قسمت المان هایی که در ادامه از آنها استفاده میکنیم را تعریف میکنیم.

```
// here we define the class of fund requests that sends the users
// request to increase funds to an admin and awaits for it
class FundRequest{
    private User user;
    private double amount;
```

متود سازنده کلاس را معرفی کرده و مقدار اولیه می دهیم.

```
    // here we define the method of the fund request constructor
    public FundRequest(User user, double amount) {
        this.user = user;
        this.amount = amount;
    }
```

متودی برای گرفتن کاربر و میزان درخواست افزایش موجودی او معرفی می کنیم.

```
    // here we define the method to get user
    public User getUser() {
        return user;
    }

    // here we define the method to get the amount
    public double getAmount() {
        return amount;
    }
```

حال کلاس اصلی Main را معرفی می کنیم.
به تابع می گوییم که ورودی خواهیم داشت و مقادیر الیه ای جهت داشتن حداقل یک ادمین، یک فروشنده، چندین محصول از فروشنده و یک کاربر خواهیم داشت.

```
// here we define the Main class to run the code
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        OnlineShop shop = new OnlineShop("DigiKala", "www.digikala.com", "123-456-
7890");

        // here we define an initial predefined admin
        Admin initialAdmin = new Admin("digi", "kala", "digi@kala.com", "123-456-
7890", "digi Tehran");
        shop.registerAdmin(initialAdmin);
```

```
        // Add a seller with 2 books and a computer
        Seller shayan = new Seller("digidigi", "shayan", "vafaei",
"shayan@vafaei.com", "913-403-7056", "Shahid Beheshti University");
        shop.registerUser(shayan);
        initialAdmin.approveSeller(shayan);

        shayan.addProduct(new Books("Book 1", 19.49, 10), shop);
        shayan.addProduct(new Books("Book 2", 24.69, 5), shop);
        shayan.addProduct(new Electronics("Computer", 979.97, 6), shop);
```

منوی اول که شامل ثبت نام، ورود و خروج از سامانه است را تعریف می کنیم.

```
                // here we try to register a customer or a seller
                case 1:
                    System.out.println("Register as:\n1. Customer\n2. Seller\n0. Back
to main menu");
                    int type = scanner.nextInt();
                    scanner.nextLine();

                    System.out.print("Enter username: ");
                    String username = scanner.nextLine();
                    System.out.print("Enter password: ");
                    String password = scanner.nextLine();
                    System.out.print("Enter email: ");
                    String email = scanner.nextLine();
                    System.out.print("Enter phone number: ");
                    String phoneNumber = scanner.nextLine();
                    System.out.print("Enter address: ");
                    String address = scanner.nextLine();

                    if (type == 1) {
                        Customer customer = new Customer(username, password, email,
phoneNumber, address);
                        shop.registerUser(customer);
                    } else if (type == 2) {
                        System.out.print("Enter company name: ");
                        String companyName = scanner.nextLine();
                        Seller seller = new Seller(companyName, username, password,
email, phoneNumber, address);
                        shop.registerUser(seller);
                    }
                    break;
```

```
                case 2:
                    System.out.println("Login as:\n1. User\n2. Seller\n3. Admin");
                    int loginType = scanner.nextInt();
                    scanner.nextLine();

                    System.out.print("Enter username: ");
                    username = scanner.nextLine();
                    System.out.print("Enter password: ");
                    password = scanner.nextLine();

                    User user = shop.loginUser(username, password);
                    if (user != null) {
                        boolean loggedIn = true;
                        while (loggedIn) {
                            if (user instanceof Admin) {
                                adminMenu((Admin) user, shop, scanner);
                            } else if (user instanceof Seller) {
                                sellerMenu((Seller) user, shop, scanner);
                            } else {
                                customerMenu((Customer) user, shop, scanner);
                            }
                            System.out.println("0. Back to main menu");
                            System.out.println("1. Logout");
                            int logoutChoice = scanner.nextInt();
                            scanner.nextLine();
                            if (logoutChoice == 0) {
                                loggedIn = false;
                            } else if (logoutChoice == 1) {
                                return;
                            }
                        }
                    }
                    break;

                case 3:
                    System.out.println("Exiting...");
                    return;

                case 0:
                    return;

                default:
                    System.out.println("Invalid choice. Try again.");
            }
        }
    }
```

حال منوی ادمین را معرفی می کنیم.

```
    private static void adminMenu(Admin admin, OnlineShop shop, Scanner scanner) {
        System.out.println("1. View Profile");
        System.out.println("2. Register New Admin");
        System.out.println("3. Approve Seller");
        System.out.println("4. View Fund Requests");
        System.out.println("5. Approve Fund Request");
        System.out.println("0. Logout");

        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 0:
               System.out.println("Logging out...");
               return;
            case 1:
                admin.viewProfile();
                break;
            case 2:
                System.out.print("Enter new admin username: ");
                String username = scanner.nextLine();
                System.out.print("Enter new admin password: ");
                String password = scanner.nextLine();
                System.out.print("Enter new admin email: ");
                String email = scanner.nextLine();
                System.out.print("Enter new admin phone number: ");
                String phoneNumber = scanner.nextLine();
                System.out.print("Enter new admin address: ");
                String address = scanner.nextLine();
                Admin newAdmin = new Admin(username, password, email, phoneNumber,
address);
                admin.registerNewAdmin(shop, newAdmin);
                break;
            case 3:
                approveSeller(admin, shop, scanner);
                break;
            case 4:
                admin.viewFundRequests();
                break;
            case 5:
                System.out.print("Enter the index of the fund request to approve: ");
                int index = scanner.nextInt() - 1;
                admin.approveFundRequest(index);
                break;
```

```
            default:
                System.out.println("Invalid choice.");
        }
    }
```

حال منوی فروشنده را تعریف می کنیم.

```
    private static void sellerMenu(Seller seller, OnlineShop shop, Scanner scanner) {
        System.out.println("1. View Profile");
        System.out.println("2. Add Product");
        System.out.println("3. View Available Products");
        System.out.println("0. Logout");

        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 0:
               System.out.println("Logging out...");
               return;
            case 1:
                seller.viewProfile();
                break;
            case 2:
                System.out.println("Select product category:");
                System.out.println("1. Electronics");
                System.out.println("2. Books");
                System.out.println("3. Clothing");
                System.out.println("4. Jewelry");
                System.out.println("5. Cars");
                int category = scanner.nextInt();
                scanner.nextLine();

                System.out.print("Enter product name: ");
                String name = scanner.nextLine();
                System.out.print("Enter product price: ");
                double price = scanner.nextDouble();
                System.out.print("Enter product inventory: ");
                int inventory = scanner.nextInt();

                Product newProduct;
                switch (category) {
                    case 1:
                        newProduct = new Electronics(name, price, inventory);
```

```
                    break;
                case 2:
                    newProduct = new Books(name, price, inventory);
                    break;
                case 3:
                    newProduct = new Clothing(name, price, inventory);
                    break;
                case 4:
                    newProduct = new Jewelry(name, price, inventory);
                    break;
                case 5:
                    newProduct = new Cars(name, price, inventory);
                    break;
                default:
                    System.out.println("Invalid category.");
                    return;
            }
            seller.addProduct(newProduct, shop);
            break;
        case 3:
            shop.listProductsWithDetails();
            break;
        default:
            System.out.println("Invalid choice.");
    }
}
```

حال منوی مشتری را معرفی می کنیم.

```
    private static void customerMenu(Customer customer, OnlineShop shop, Scanner
scanner) {
        while (true) {
            System.out.println("1. View Profile");
            System.out.println("2. View Cart");
            System.out.println("3. Request Funds");
            System.out.println("4. Add Product to Cart");
            System.out.println("5. Place Order");
            System.out.println("6. View Available Products");
            System.out.println("7. Add review");
            System.out.println("0. Back to Main Menu");

            int choice = scanner.nextInt();
            scanner.nextLine();
```

```
            switch (choice) {
                case 0:
                    System.out.println("Logging out...");
                    return;
                case 1:
                    customer.viewProfile();
                    break;
                case 2:
                    customer.viewCart();
                    break;
                case 3:
                    System.out.print("Enter amount to request: ");
                    double amount = scanner.nextDouble();
                    customer.requestFunds(amount, getAdmins(shop));
                    break;
                case 4:
                    addProductToCart(customer, shop, scanner);
                    break;
                case 5:
                    if (customer.placeOrder(shop)) {
                        System.out.println("Order placed successfully.");
                    } else {
                        System.out.println("Failed to place order. Please check your
cart and wallet balance.");
                    }
                    break;
                case 6:
                    shop.listProductsWithDetails();
                    break;
                case 7:
                    addReview(customer, shop, scanner);
                    break;
                default:
                    System.out.println("Invalid choice.");
            }
        }
    }
```

حال حلقه نظرات کاربران از محصولات را معرفی می کنیم.

```
    private static void addReview(Customer customer, OnlineShop shop, Scanner scanner)
{
        shop.listProductsWithDetails();
        System.out.print("Enter the index of the product you want to review: ");
        int productIndex = scanner.nextInt() - 1;
        scanner.nextLine();

        if (productIndex >= 0 && productIndex < shop.getProducts().size()) {
            Product selectedProduct = shop.getProducts().get(productIndex);

            System.out.print("Enter your review comment: ");
            String comment = scanner.nextLine();
            System.out.print("Enter your rating (1-5): ");
            int rating = scanner.nextInt();
            scanner.nextLine();

            Review review = new Review(customer.getUsername(), comment, rating);
            selectedProduct.addReview(review);
        } else {
            System.out.println("Invalid product index.");
        }
    }
```

تابع گرفتن ادمین ها را معرفی می کنیم.

```
    private static List<Admin> getAdmins(OnlineShop shop) {
        return shop.getUsers().stream()
                .filter(user -> user instanceof Admin)
                .map(user -> (Admin) user)
                .collect(Collectors.toList());
    }
```

حال تابع اضافه کردن محصولات به سبد خرید را تعریف می کنیم.

```
private static void addProductToCart(Customer customer, OnlineShop shop, Scanner
scanner) {
        shop.listProductsWithDetails();
        System.out.print("Enter the index of the product you want to add to cart: ");
        int productIndex = scanner.nextInt() - 1;
        scanner.nextLine();


        if (productIndex >= 0 && productIndex < shop.getProducts().size()) {
            System.out.print("Enter the quantity: ");
            int quantity = scanner.nextInt();
            scanner.nextLine();

            Product selectedProduct = shop.getProducts().get(productIndex);
            customer.addToCart(selectedProduct, quantity);
        } else {
            System.out.println("Invalid product index.");
        }

    }
```

در اخر حلقه تایید فروشنده را تعریف می کنیم.

```
    private static void approveSeller(Admin admin, OnlineShop shop, Scanner scanner) {
        List<User> users = shop.getUsers();
        int index = 1;
        for (User user : users) {
            if (user instanceof Seller && !((Seller) user).isApproved()) {
                System.out.println(index + ". " + user.getUsername());
                index++;
            }
        }

        if (index == 1) {
            System.out.println("No sellers pending approval.");
            return;
        }

        System.out.print("Enter the index of the seller to approve: ");
        int sellerIndex = scanner.nextInt() - 1;
        scanner.nextLine();
```

```
        if (sellerIndex >= 0 && sellerIndex < users.size()) {
            User selectedUser = users.get(sellerIndex);
            if (selectedUser instanceof Seller) {
                admin.approveSeller((Seller) selectedUser);
            } else {
                System.out.println("Invalid user type selected.");
            }
        } else {
            System.out.println("Invalid index.");
        }
    }
```

چالش برانگیزترین قسمت پروژه نوشتن حلقه ها و ساخت منو ها بود.
همچنین پروژه های بزرگ دیباگ کردن دشوار تری نسبت به کد های کوچک تر دارند.
بزرگ بودن حلقه ران کردن فروشگاه پیچیدگی زیادی داشت که زمان زیادتری نسبت به بقیه قسمت ها گرفت.

با تشکر و احترام،
شایان وفایی