Object Oriented Programming

E-Commerce Application Project

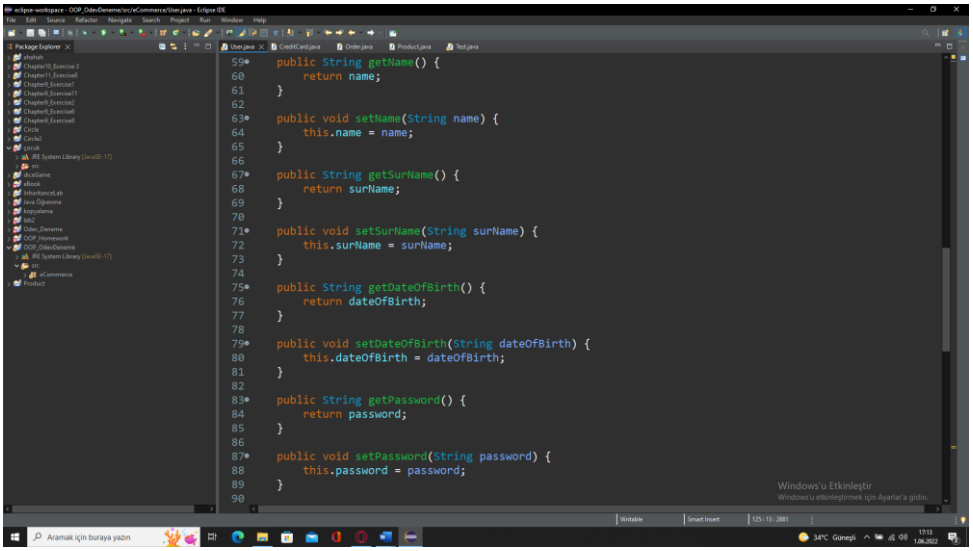
User Class:

User class holds the information about the user that is created in the app. That class has getter and setter method for our fields and arraylists.

```
1 package eCommerce;
2 import java.util.ArrayList;
3
4 public class User {
5     private String userName;
6     private String name;
7     private String surName;
8     private String dateOfBirth;
9     private String password;
0     private String eMail;
1     private String homeAdress;
2     private String workAdress;
3     private ArrayList<Product> favoriteProducts = new ArrayList<Product>();
4     private ArrayList<Product> productOrdered = new ArrayList<Product>();
5     private ArrayList<CreditCard> creditCards = new ArrayList<CreditCard>();
6
7•    public User(String userName,String name,String surName,String dateOfBirth,String password,String eMail
8             String homeAdress,String workAdress) {
9         this.userName = userName;
0         this.name = name;
1         this.surName = surName;
2         this.dateOfBirth = dateOfBirth;
3         this.password = password;
4         this.eMail = eMail;
5         this.homeAdress = homeAdress;
6         this.workAdress = workAdress;
```

In this part, we have created our fields and ArrayLists. Also we have created our constructor for user class.

```
public void setFavoriteProducts(ArrayList<Product> favoriteProducts) {
    this.favoriteProducts = favoriteProducts;
}
public ArrayList<Product> getFavoriteProducts(){
    return this.favoriteProducts;
}
public void setProductOrdered(ArrayList<Product> productOrdered) {
    this.productOrdered = productOrdered;
}
public ArrayList<Product> getProductOrdered(){
    return this.productOrdered;
}

public void setCreditCards(ArrayList<CreditCard> creditCards) {
    this.creditCards = creditCards;
}

public ArrayList<CreditCard> getCreditCards(){
    return this.creditCards;
}



public String getUserName() {
    return userName;
}

public void setUserName(String userName) {
    this.userName = userName;
}
```



```java
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getSurName() {
    return surName;
}

public void setSurName(String surName) {
    this.surName = surName;
}

public String getDateOfBirth() {
    return dateOfBirth;
}

public void setDateOfBirth(String dateOfBirth) {
    this.dateOfBirth = dateOfBirth;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}
```

```java
public String geteMail() {
    return eMail;
}

public void seteMail(String eMail) {
    this.eMail = eMail;
}

public String getHomeAdress() {
    return homeAdress;
}

public void setHomeAdress(String homeAdress) {
    this.homeAdress = homeAdress;
}

public String getWorkAdress() {
    return workAdress;
}

public void setWorkAdress(String workAdress) {
    this.workAdress = workAdress;
}

public void favoriteList(Product product) {
    this.favoriteProducts.add(product);
}
```

In above, as you can see we have created our getter and setter methods for User class.

```java
public void purchase(ArrayList<Product> productList) {
    for(int i=0;i<productList.size();i++) {
        if(productList.get(i).getStockInfo()<0) {
            productList.remove(i);
        }


        else
            System.out.println("Purchase is successful");
        break;

    }
}
```

In the above method, we have created a method to print if purchase is successful or not. If purchase is succesfull it prints “Purchase is successful” if it is not then it checks for a product stock. If product stock is smaller than 0, then it removes the item from the product list.

Credit Card Class:

In this class, we are wanted to hold information about credit card. Such as, card number, card user, security code and expiration date. Also, we need to write getter and setter methods of credit card class.

```java
package eCommerce;

public class CreditCard {
    private long cardNumber;
    private String cardUser;
    private String securityCode;
    private String expirationDate;

    public CreditCard(long cardNumber, String cardUser, String securityCode, String expirationDate) {
        this.cardNumber = cardNumber;
        this.cardUser = cardUser;
        this.securityCode = securityCode;
        this.expirationDate = expirationDate;
```

In the above, we created our fields and constructor.

```
public long getCardNumber() {
    return this.cardNumber;
}
public void setCardNumber(long cardNumber) {
    this.cardNumber = cardNumber;
}
public String getCardUser() {
    return this.cardUser;
}
public void setCardUser(String cardUser) {
    this.cardUser = cardUser;
}
public String getSecurityCode() {
    return this.securityCode;
}
public void setSecurityCode(String securityCode) {
    this.securityCode = securityCode;
}
public String getExpirationDate() {
    return this.expirationDate;
}
public void setExpirationDate(String expirationDate) {
    this.expirationDate = expirationDate;
}
```

And above you can see the getter and setter methods.

Order Class:

In order class, we are wanted to hold information about ordering user object, the ordered product object and the credit card objects to which is payment made.

```
1 package eCommerce;
2
3 public class Order {
4     private User user;
5     private Product product;
6     private CreditCard creditCard;
7
8     public Order(User user,Product product,CreditCard creditCard) {
9         this.user = user;
0         this.product = product;
1         this.creditCard = creditCard;
2     }
```

In here, we created our fields and constructor for Order class.

```java
public User getUser() {
    return this.user;
}

public void setUser(User user) {
    this.user = user;
}

public Product getProduct() {
    return this.product;
}

public void setProduct(Product product) {
    this.product = product;
}

public CreditCard getCreditCard() {
    return this.creditCard;
}

public void setCreditCard(CreditCard creditCard) {
    this.creditCard = creditCard;
}
```

In here, as you can see I wrote the getter and setter methods of Order class.

Product Class:

In product class, we are wanted to create fields that hold product name, product color, product category, product stock information, product weight, product description information. Also we are wanted to decrease the number of stocks as much as the number of products purchased and controls stock number.

```java
package eCommerce;

public class Product {
    private String productName;
    private String productColor;
    private String productCategory;
    private int stockInfo;
    private int productWeight;
    private String description;

    public Product(String productName,String productColor,String productCategory,int stockInfo,
            int productWeight,String description) {
        this.productName = productName;
        this.productColor = productColor;
        this.productCategory = productCategory;
        this.stockInfo = stockInfo;
        this.productWeight = productWeight;
        this.description = description;
    }
```

In above, we created our fields and constructor of product class.

```java
    public String getProductName() {
        return this.productName;
    }

    public void setProductName(String productName) {
        this.productName = productName;
    }

    public String getProductColor() {
        return this.productColor;
    }

    public void setProductColor(String productColor) {
        this.productColor = productColor;
    }

    public String getProductCategory() {
        return this.productCategory;
    }

    public void setProductCategory(String productCategory) {
        this.productCategory = productCategory;
    }

    public int getStockInfo() {
        return this.stockInfo;
    }

    public void setStockInfo(int stockInfo) {
        this.stockInfo = stockInfo;
    }
```

```java
public int getProductWeight() {
    return this.productWeight;
}

public void setProductWeight(int productWeight) {
    this.productWeight = productWeight;
}

public String getDescription() {
    return this.description;
}

public void setDescription(String description) {
    this.description = description;
}
```

In the above parts, we have written getter and setter methods of Product class.

```java
public int reducingStock(int number) {
    this.stockInfo = this.getStockInfo()-number;
    if(this.stockInfo<0) {
        System.out.println("There is no stock!");
        return this.stockInfo;
    }
    else
        return this.stockInfo;
}
```

In reducingStock method, I aimed to return a stock information after user purchase. If there is not enough stock. Then program is going to print a error message that says “There is no stock!”.

Test Class:

In test class, we are wanted to check operations. Such as, adding users, adding credit cards, adding products, purchasing methods and favoring products.

```java
Scanner input = new Scanner(System.in);
static ArrayList<Product> productList = new ArrayList<Product>();
public void addingProducts(Product product){
    productList.add(product);
}

public void displayProducts() {
    for(int i = 0; i<productList.size();i++) {
        System.out.println("Product name: "+productList.get(i).getProductName()+"\n"+"Product Color: "+
                "Product Category: "+productList.get(i).getProductCategory()+"\n"+"Product stock: "+pro
                "Product Weight: "+productList.get(i).getProductWeight()+" g\n"+"Product description: "
                "----------------------------");
    }
```

In above part, I have create an object to get an input from user. Then I created an ArrayList that has a product type. After that, I wrote a method to add product to list.

Lastly, I have display the products on the screen to let user to choose the products.

```
public  void productControlAndPurchasing(User u,String answer){
    for(Product p:productList){
        if(answer.equals(p.getProductName())){
            u.getProductOrdered().add(p);
            System.out.println("Added to ordered list. ");
            System.out.println("How many will you buy: ");
            int reply=input.nextInt();
            p.reducingStock(reply);
            u.purchase(productList);
            System.out.println("Do you want to add your favorite list (1 for yes/2 for no): ");
            int addingAnswer=input.nextInt();
            if(addingAnswer==1){
                u.favoriteList(p);
                break;
            }
            else;
                System.out.println("Item is not added to favorite list");
        }
        }
    }
```

Here, I wanted to check what product user wants to buy and if it is equals to the product that we add. I used a for loop to check product in product list. If users choice equals to the product name then it will add product to ordered product list. And the program will ask user to how many he/she wants to buy. According to their answer, program will reduce the stock. Then it will invoke the purchase method. And it will ask if he/she wants to add product to favorite list.

```
package eCommerce;

import java.util.ArrayList;

public class Test {
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        Test test = new Test();

        User user1 = new User("cmpeStudent","baran","demir","22.05.2001","123456","baran54123@gmail.com",
                "Izmir","Manisa");
        CreditCard card1 = new CreditCard(1234567891,"baran","1571","2026");
        Product product1 = new Product("nike tshirt","white","clothes",1000,50,"black oversize tshirt");
        Product product2 = new Product("nike air force","black","shoes",500,500,"sport shoes");
        test.addingProducts(product1);
        test.addingProducts(product2);
        test.displayProducts();
        while(true) {
            System.out.println("What would you like to buy: ");
            String productChoice = input.nextLine();
            test.productControlAndPurchasing(user1, productChoice);
            System.out.println("Do you want to buy more things: (if yes press 1 otherwise press 2): ");
            String reply = input.nextLine();
            int flag = Integer.parseInt(reply);
            if(flag==1) {
                continue;
            }
            else
                break;
        }
```

In here, we created a card object, and two product objects. Then I added them to list. After adding them to the list. I displayed them on the screen. And program asks user to what they want to buy. According to her/his answer program invokes the "productControlAndPurchasin" method and sends the parameters.

Also program asks the if they want to continue to shop. There is a flag variable to check if they want to continue.

```
for(int i=0;i<user1.getFavoriteProducts().size();i++) {

    System.out.println("Favorite Products: "+user1.getFavoriteProducts().get(i).getProductName());
    System.out.println("Shopping is completed");
}
```

In here, program displays the favorite products if they are added and says that shopping is completed.

OUTPUT EXAMPLE:

```
Product name: nike tshirt
Product Color: white
Product Category: clothes
Product stock: 1000
Product Weight: 50 g
Product description: black oversize tshirt
----------------------------
Product name: nike air force
Product Color: black
Product Category: shoes
Product stock: 500
Product Weight: 500 g
Product description: sport shoes
----------------------------
What would you like to buy:
nike air force
Added to ordered list.
How many will you buy:
5
Purchase is successful
Do you want to add your favorite list (1 for yes/2 for no):
1
Do you want to buy more things: (if yes press 1 otherwise press 2):
2
Favorite Products: nike air force
Shopping is completed
```

Baran Çakmak Demir 210315002