MANİSA
CELAL BAYAR
ÜNİVERSİTESİ

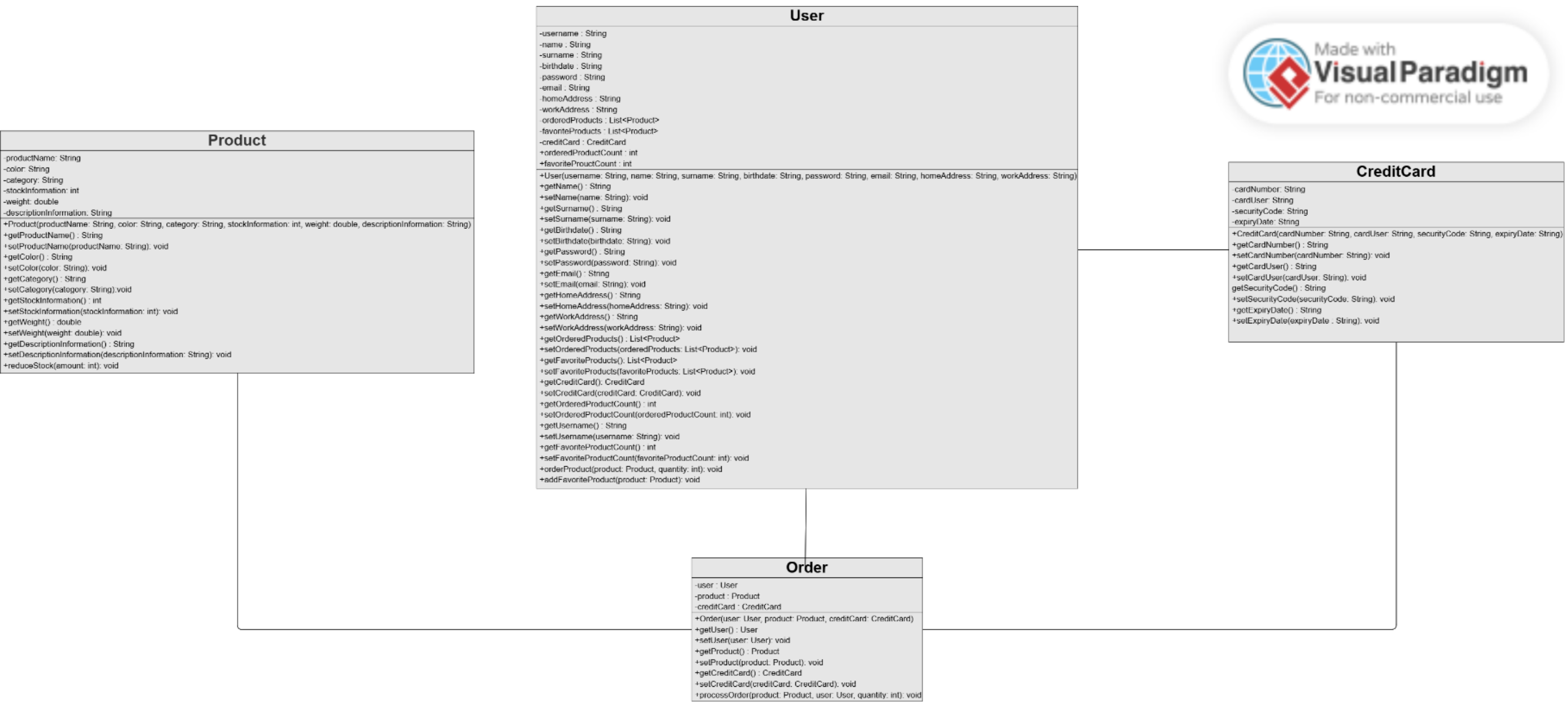
OBJECT ORİENTED PROGRAMMİNG

E-Commerce Application Project

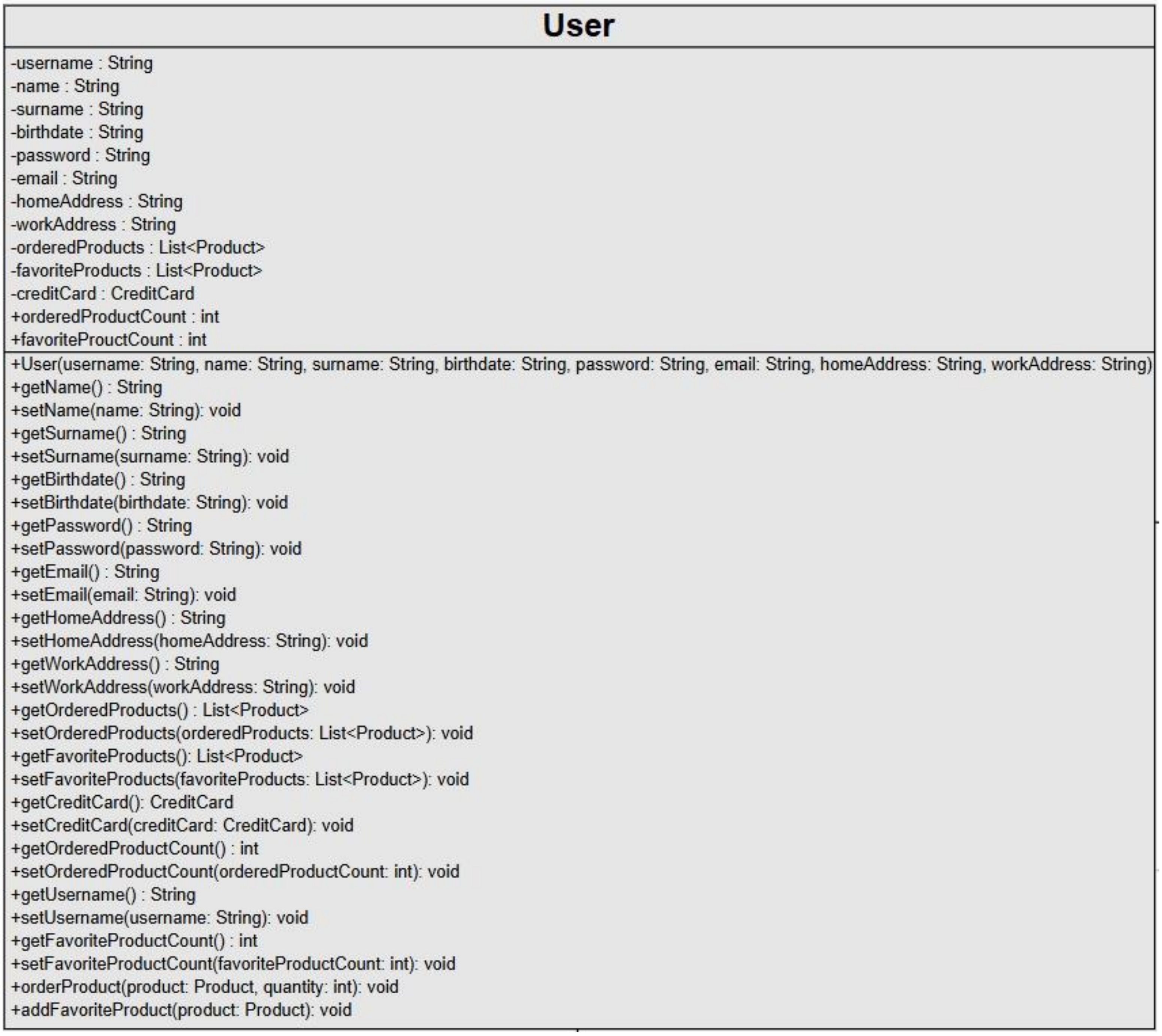
220315040 -- Emine ÇETİN

220315082 -- Mustafa Furkan YILMAZ

UML DİAGRAM


User
-username : String
-name : String
-surname : String
-birthdate : String
-password : String
-email : String
-homeAddress : String
-workAddress : String
-orderedProducts : List<Product>
-favoriteProducts : List<Product>
-creditCard : CreditCard
+orderedProductCount : int
+favoriteProuctCount : int
+User(username: String, name: String, surname: String, birthdate: String, password: String, email: String, homeAddress: String, workAddress: String)
+getName() : String
+setName(name: String): void
+getSurname() : String
+setSurname(surname: String): void
+getBirthdate() : String
+setBirthdate(birthdate: String): void
+getPassword() : String
+setPassword(password: String): void
+getEmail() : String
+setEmail(email: String): void
+getHomeAddress() : String
+setHomeAddress(homeAddress: String): void
+getWorkAddress() : String
+setWorkAddress(workAddress: String): void
+getOrderedProducts() : List<Product>
+setOrderedProducts(orderedProducts: List<Product>): void
+getFavoriteProducts(): List<Product>
+setFavoriteProducts(favoriteProducts: List<Product>): void
+getCreditCard(): CreditCard
+setCreditCard(creditCard: CreditCard): void
+getOrderedProductCount() : int
+setOrderedProductCount(orderedProductCount: int): void
+getUsername() : String
+setUsername(username: String): void
+getFavoriteProductCount() : int
+setFavoriteProductCount(favoriteProductCount: int): void
+orderProduct(product: Product, quantity: int): void
+addFavoriteProduct(product: Product): void
Made with Visual Paradigm For non-commercial use
Product
-productName: String
-color: String
-category: String
-stockInformation: int
-weight: double
-descriptionInformation: String
+Product(productName: String, color: String, category: String, stockInformation: int, weight: double, descriptionInformation: String)
+getProductName() : String
+setProductName(productName: String): void
+getColor() : String
+setColor(color: String): void
+getCategory() : String
+setCategory(category: String):void
+getStockInformation() : int
+setStockInformation(stockInformation: int): void
+getWeight() : double
+setWeight(weight: double): void
+getDescriptionInformation() : String
+setDescriptionInformation(descriptionInformation: String): void
+reduceStock(amount: int): void
CreditCard
-cardNumber: String
-cardUser: String
-securityCode: String
-expiryDate: String
+CreditCard(cardNumber: String, cardUser: String, securityCode: String, expiryDate: String)
+getCardNumber() : String
+setCardNumber(cardNumber: String): void
+getCardUser() : String
+setCardUser(cardUser: String): void
getSecurityCode() : String
+setSecurityCode(securityCode: String): void
+getExpiryDate() : String
+setExpiryDate(expiryDate : String): void
Order
-user : User
-product : Product
-creditCard : CreditCard
+Order(user: User, product: Product, creditCard: CreditCard)
+getUser() : User
+setUser(user: User): void
+getProduct() : Product
+setProduct(product: Product): void
+getCreditCard() : CreditCard
+setCreditCard(creditCard: CreditCard): void
+processOrder(product: Product, user: User, quantity: int): void

USER CLASS



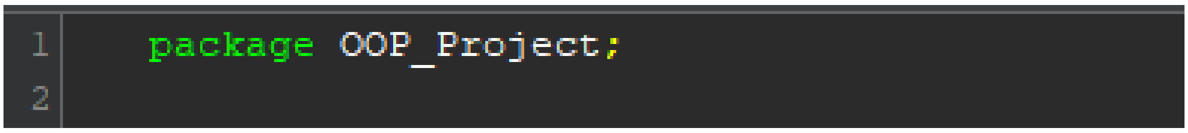
User

-username : String
-name : String
-surname : String
-birthdate : String
-password : String
-email : String
-homeAddress : String
-workAddress : String
-orderedProducts : List<Product>
-favoriteProducts : List<Product>
-creditCard : CreditCard
+orderedProductCount : int
+favoriteProuctCount : int

+User(username: String, name: String, surname: String, birthdate: String, password: String, email: String, homeAddress: String, workAddress: String)
+getName() : String
+setName(name: String): void
+getSurname() : String
+setSurname(surname: String): void
+getBirthdate() : String
+setBirthdate(birthdate: String): void
+getPassword() : String
+setPassword(password: String): void
+getEmail() : String
+setEmail(email: String): void
+getHomeAddress() : String
+setHomeAddress(homeAddress: String): void
+getWorkAddress() : String
+setWorkAddress(workAddress: String): void
+getOrderedProducts() : List<Product>
+setOrderedProducts(orderedProducts: List<Product>): void
+getFavoriteProducts(): List<Product>
+setFavoriteProducts(favoriteProducts: List<Product>): void
+getCreditCard(): CreditCard
+setCreditCard(creditCard: CreditCard): void
+getOrderedProductCount() : int
+setOrderedProductCount(orderedProductCount: int): void
+getUsername() : String
+setUsername(username: String): void
+getFavoriteProductCount() : int
+setFavoriteProductCount(favoriteProductCount: int): void
+orderProduct(product: Product, quantity: int): void
+addFavoriteProduct(product: Product): void

This class is used to store a user's information and the products they have ordered and marked as favorites. Getter and setter methods provide access to and modification of this information. The class also includes methods for ordering products and adding favorite products.

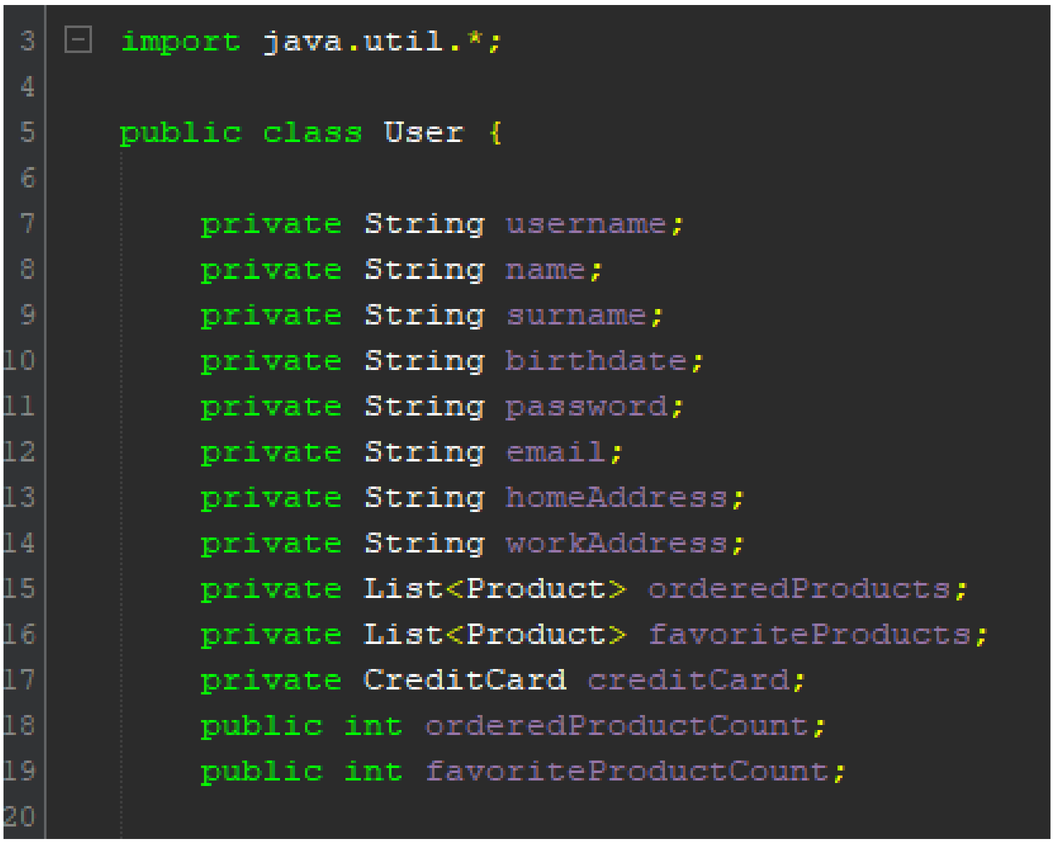
1. PACKAGE DECLARATION:



```java
package OOP_Project;

```

This line specifies that the class is part of the “OOP_Project “package. In Java, packages are used to organize classes and prevent naming conflicts.

2. User Class Definition and Fields:

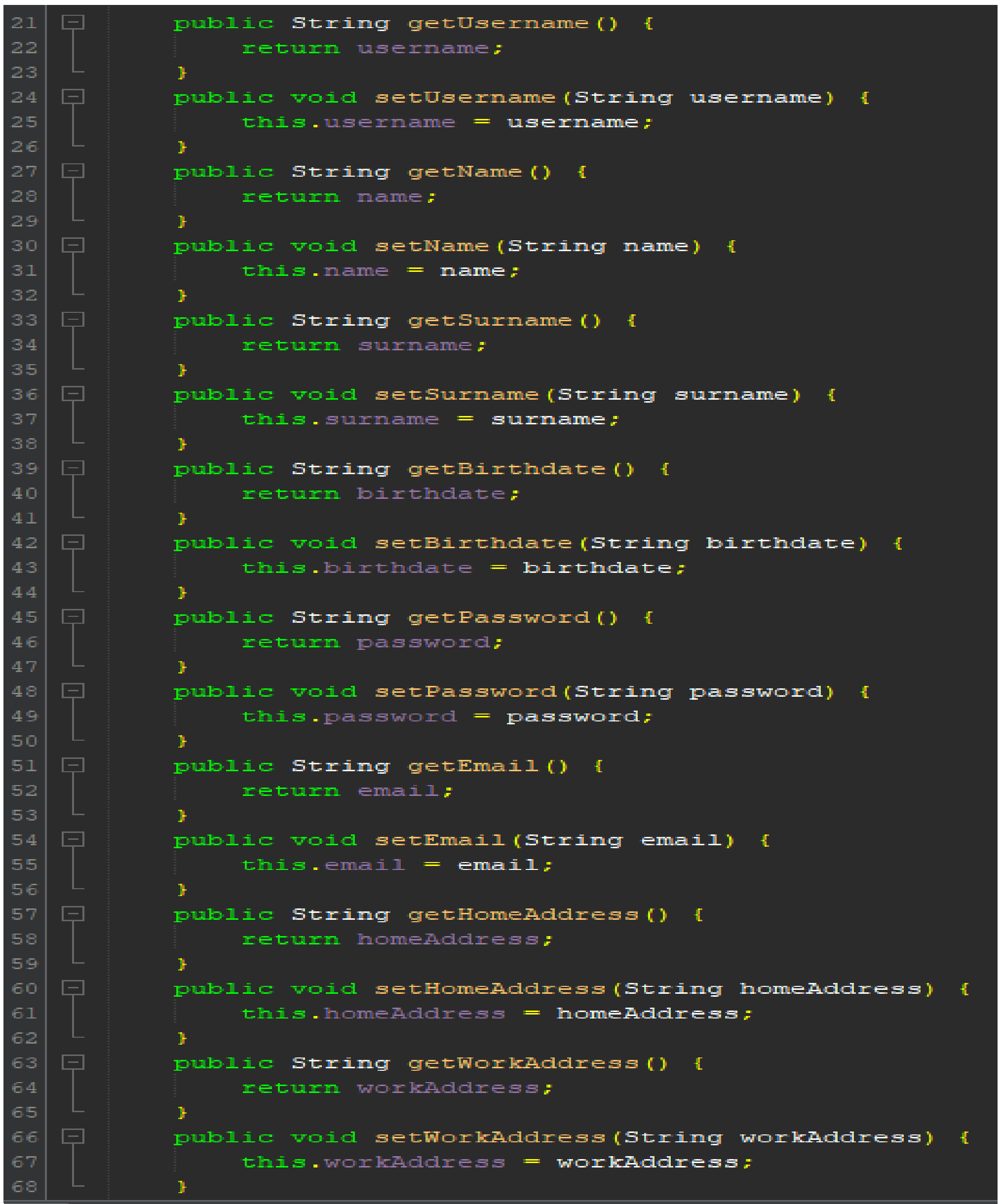


```java
import java.util.*;

public class User {

    private String username;
    private String name;
    private String surname;
    private String birthdate;
    private String password;
    private String email;
    private String homeAddress;
    private String workAddress;
    private List<Product> orderedProducts;
    private List<Product> favoriteProducts;
    private CreditCard creditCard;
    public int orderedProductCount;
    public int favoriteProductCount;

```

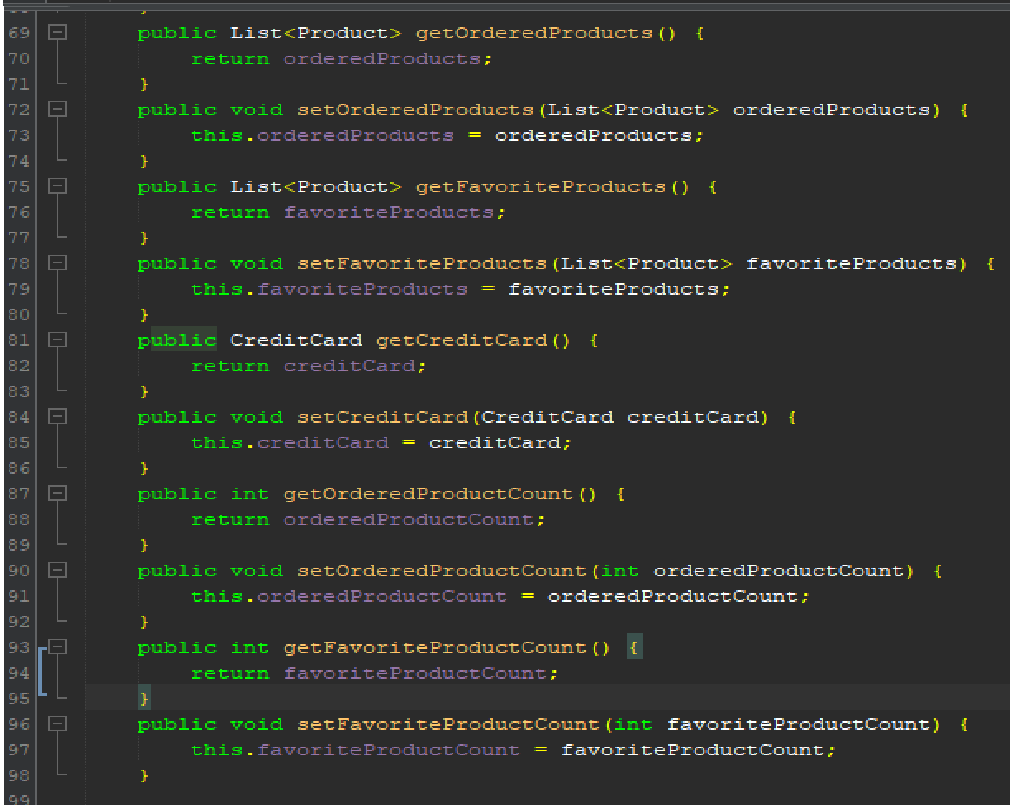
This part starts the definition of the User class and includes various private (private) and public (public) fields:

- username, name, surname, birthdate, password, email, homeAddress, workAddress: Fields that store basic user information.
- orderedProducts: A list that stores the products ordered by the user.
- favoriteProducts: A list that stores the products marked as favorites by the user.
- creditCard: Stores the user's credit card information.
- orderedProductCount, favoriteProductCount: Public fields that store the count of ordered and favorite products, respectively.

3. Getter and Setter Methods:



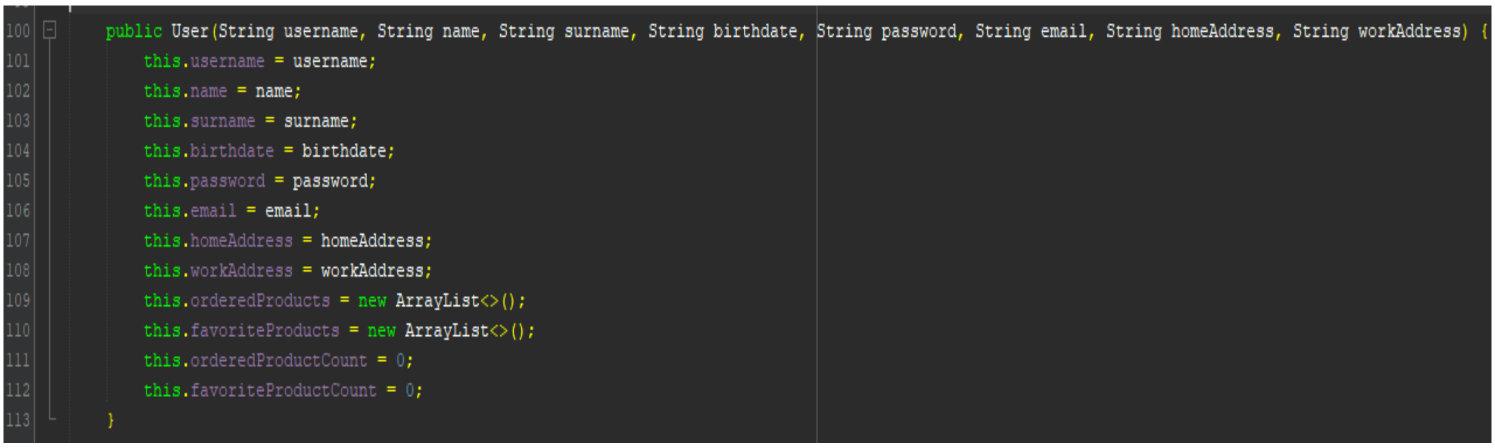
```java
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getSurname() {
        return surname;
    }
    public void setSurname(String surname) {
        this.surname = surname;
    }
    public String getBirthdate() {
        return birthdate;
    }
    public void setBirthdate(String birthdate) {
        this.birthdate = birthdate;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getHomeAddress() {
        return homeAddress;
    }
    public void setHomeAddress(String homeAddress) {
        this.homeAddress = homeAddress;
    }
    public String getWorkAddress() {
        return workAddress;
    }
    public void setWorkAddress(String workAddress) {
        this.workAddress = workAddress;
    }
```



```java
    public List<Product> getOrderedProducts() {
        return orderedProducts;
    }
    public void setOrderedProducts(List<Product> orderedProducts) {
        this.orderedProducts = orderedProducts;
    }
    public List<Product> getFavoriteProducts() {
        return favoriteProducts;
    }
    public void setFavoriteProducts(List<Product> favoriteProducts) {
        this.favoriteProducts = favoriteProducts;
    }
    public CreditCard getCreditCard() {
        return creditCard;
    }
    public void setCreditCard(CreditCard creditCard) {
        this.creditCard = creditCard;
    }
    public int getOrderedProductCount() {
        return orderedProductCount;
    }
    public void setOrderedProductCount(int orderedProductCount) {
        this.orderedProductCount = orderedProductCount;
    }
    public int getFavoriteProductCount() {
        return favoriteProductCount;
    }
    public void setFavoriteProductCount(int favoriteProductCount) {
        this.favoriteProductCount = favoriteProductCount;
    }
```

This section defines getter and setter methods for the class fields. These methods allow access to and modification of the fields. For example, the `getUsername` method returns the value of the `username` field, while the `setUsername` method updates this field with a new value.

4. Constructor Methods:



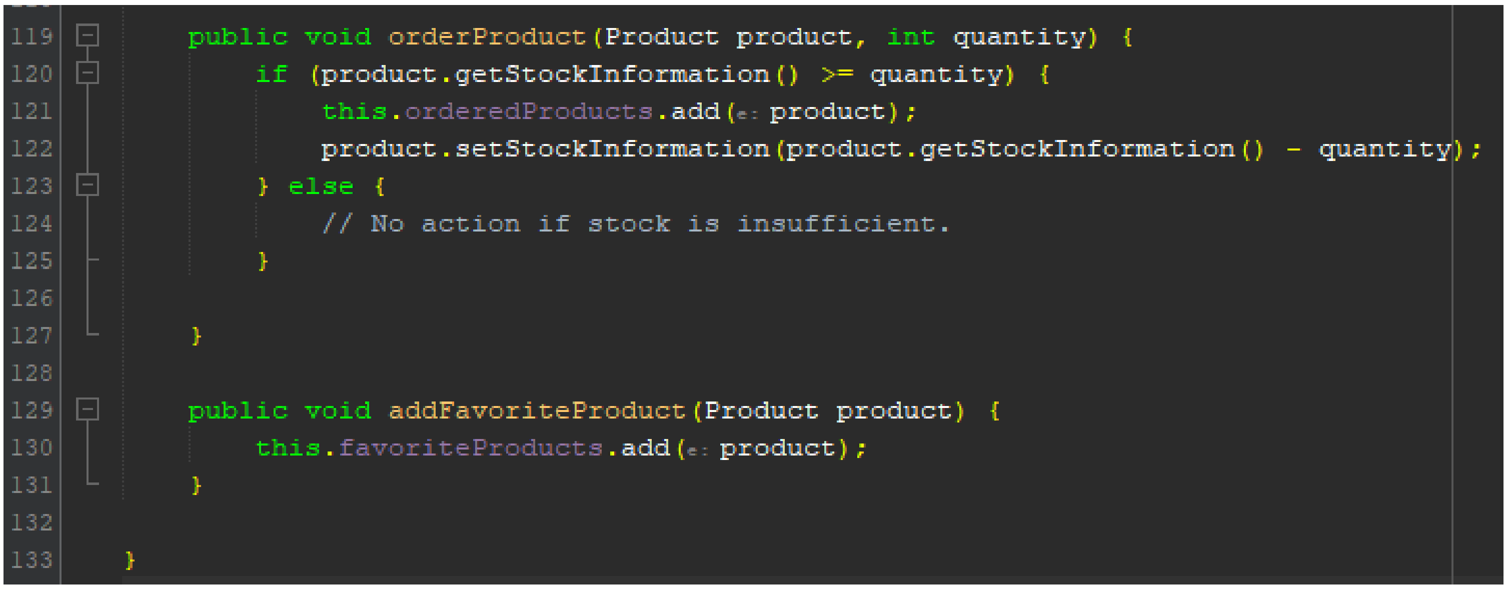
```java
    public User(String username, String name, String surname, String birthdate, String password, String email, String homeAddress, String workAddress) {
        this.username = username;
        this.name = name;
        this.surname = surname;
        this.birthdate = birthdate;
        this.password = password;
        this.email = email;
        this.homeAddress = homeAddress;
        this.workAddress = workAddress;
        this.orderedProducts = new ArrayList<>();
        this.favoriteProducts = new ArrayList<>();
        this.orderedProductCount = 0;
        this.favoriteProductCount = 0;
    }
```

This section defines two constructor methods:

The no-argument constructor: Does not perform any actions; it is a default constructor.

The parameterized constructor: Takes basic user information as parameters and initializes the corresponding fields. It also initializes the orderedProducts and favoriteProducts lists and sets the product counters to zero.

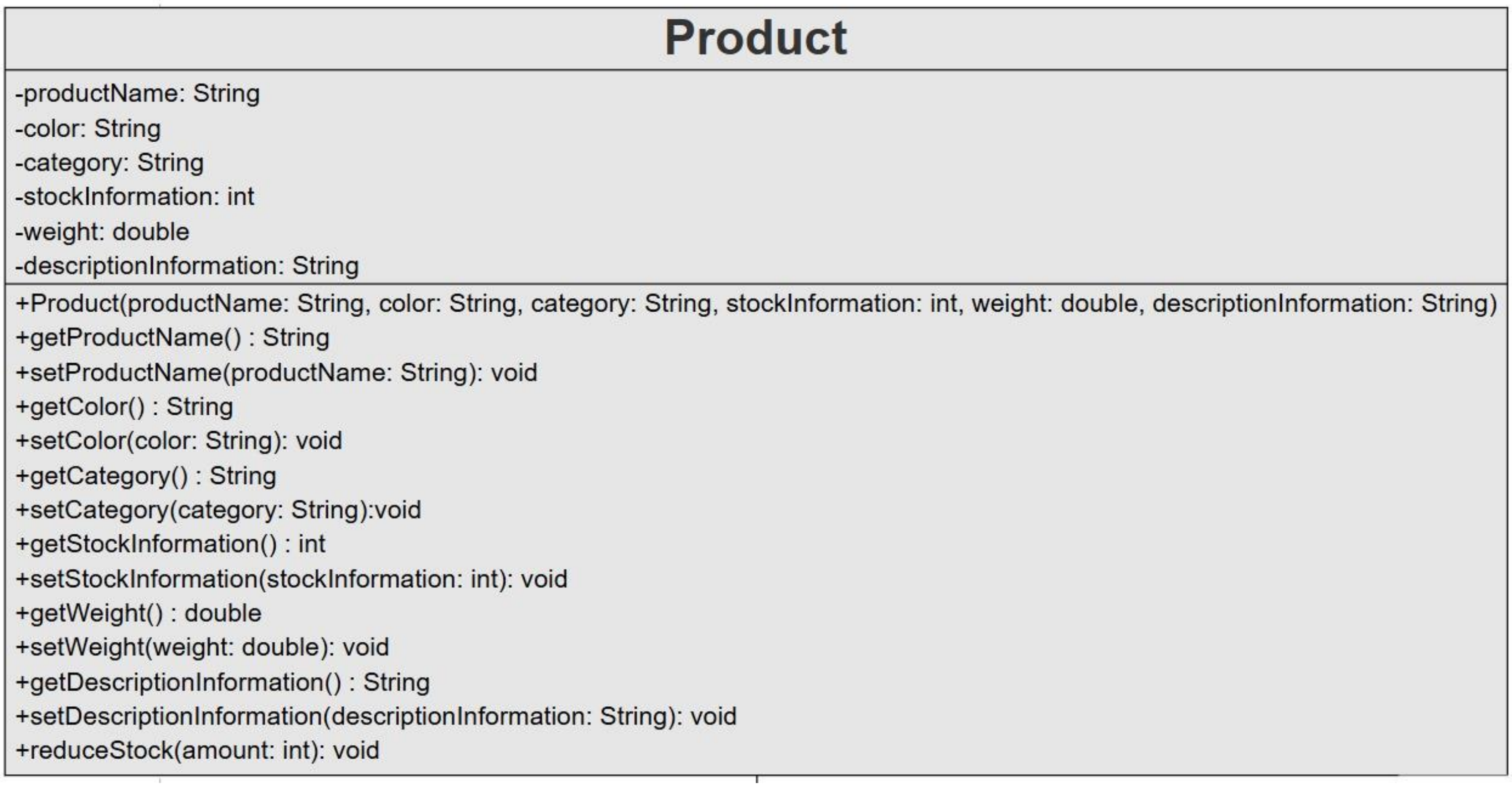
5. orderProduct and addFavoriteProduct Methods:



```java
    public void orderProduct(Product product, int quantity) {
        if (product.getStockInformation() >= quantity) {
            this.orderedProducts.add(e: product);
            product.setStockInformation(product.getStockInformation() - quantity);
        } else {
            // No action if stock is insufficient.
        }

    }

    public void addFavoriteProduct(Product product) {
        this.favoriteProducts.add(e: product);
    }

}
```

- **orderProduct:** This method takes a product and a quantity. If the product's stock information is sufficient, the product is added to the orderedProducts list, and the product's stock information is updated.
- **addFavoriteProduct**: This method adds a given product to the favoriteProducts list.

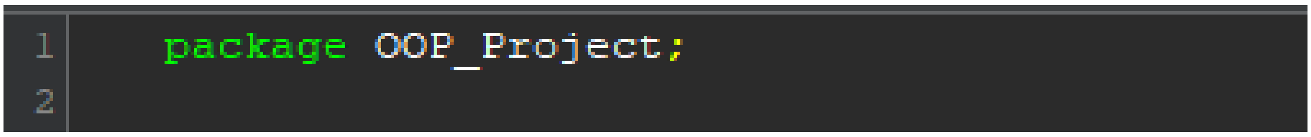
PRODUCT CLASS



```
Product
-productName: String
-color: String
-category: String
-stockInformation: int
-weight: double
-descriptionInformation: String
+Product(productName: String, color: String, category: String, stockInformation: int, weight: double, descriptionInformation: String)
+getProductName() : String
+setProductName(productName: String): void
+getColor() : String
+setColor(color: String): void
+getCategory() : String
+setCategory(category: String):void
+getStockInformation() : int
+setStockInformation(stockInformation: int): void
+getWeight() : double
+setWeight(weight: double): void
+getDescriptionInformation() : String
+setDescriptionInformation(descriptionInformation: String): void
+reduceStock(amount: int): void
```

This code defines a Product class in Java that represents a product with various attributes such as name, color, category, stock information, weight, and description. It provides methods to access and modify these attributes and includes a method to reduce the stock of the product. Let's explain each part of the code step by step:

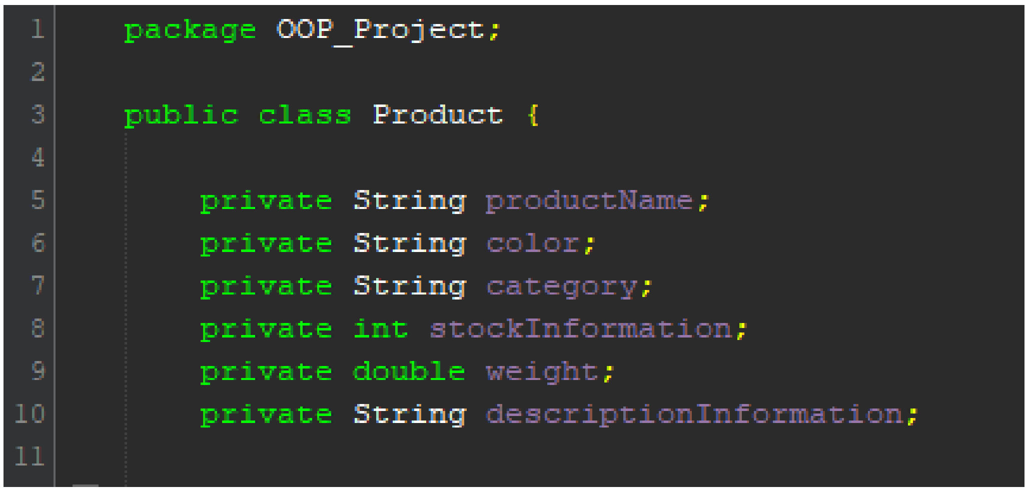
1. Package Declaration:



```java
package OOP_Project;

```

This line specifies that the class is part of the OOP_Project package. In Java, packages are used to organize classes and prevent naming conflicts.

2. Product Class Definition and Fields:

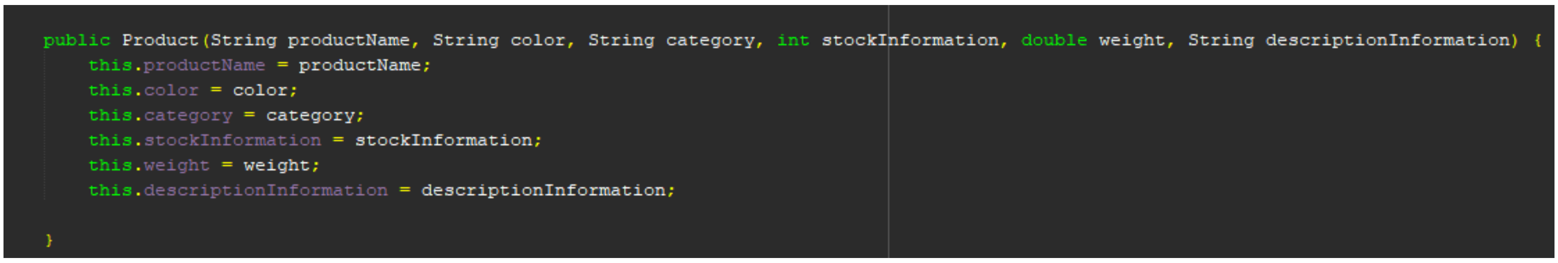


```java
package OOP_Project;

public class Product {

    private String productName;
    private String color;
    private String category;
    private int stockInformation;
    private double weight;
    private String descriptionInformation;

```

This part starts the definition of the Product class and includes several private fields:

- productName: The name of the product.
- color: The color of the product.
- category: The category to which the product belongs.
- stockInformation: The quantity of the product available in stock.
- weight: The weight of the product.
- descriptionInformation: Additional descriptive information about the product.

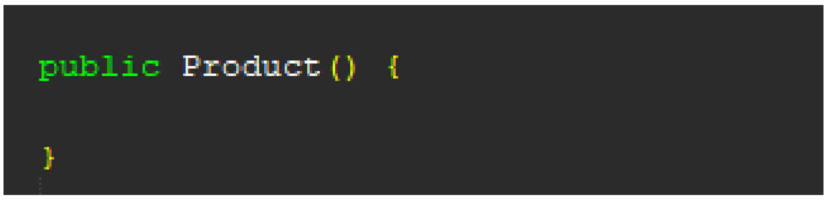
3. Constructor Methods:



```java
public Product(String productName, String color, String category, int stockInformation, double weight, String descriptionInformation) {
    this.productName = productName;
    this.color = color;
    this.category = category;
    this.stockInformation = stockInformation;
    this.weight = weight;
    this.descriptionInformation = descriptionInformation;

}
```

- This is a parameterized constructor that initializes the Product object with the given values for productName, color, category, stockInformation, weight, and descriptionInformation.

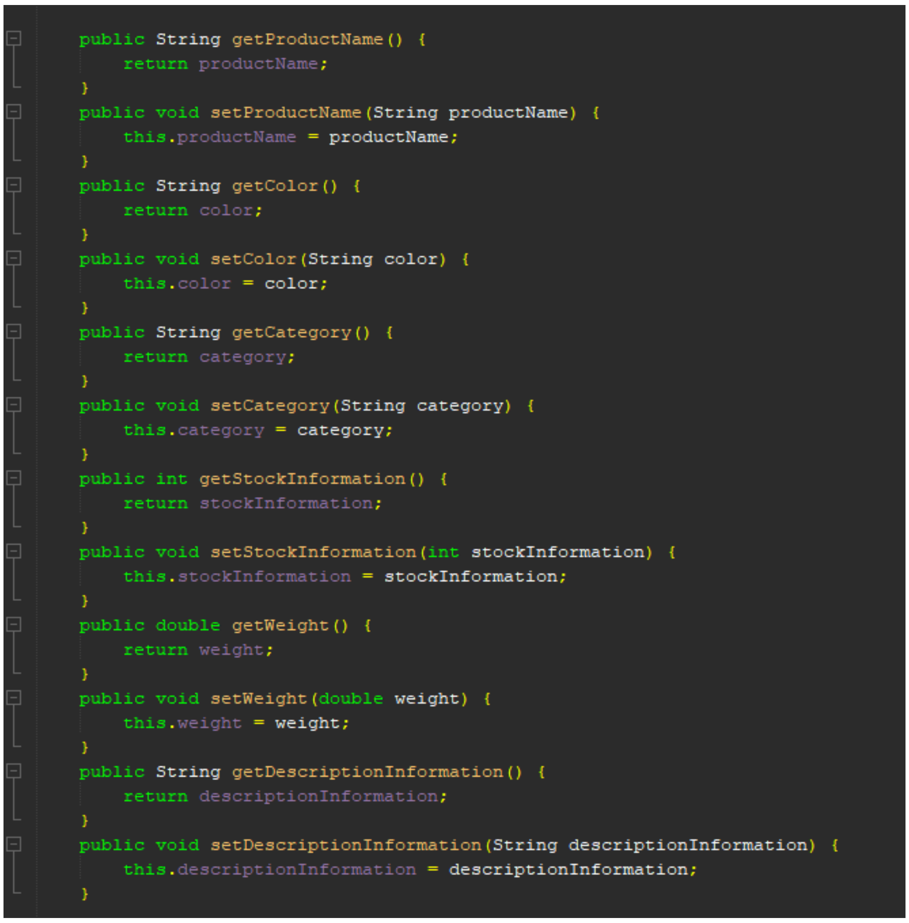
4. Default Constructor:



```java
public Product() {

}
```

This is a no-argument constructor that does not perform any actions. It is a default constructor.

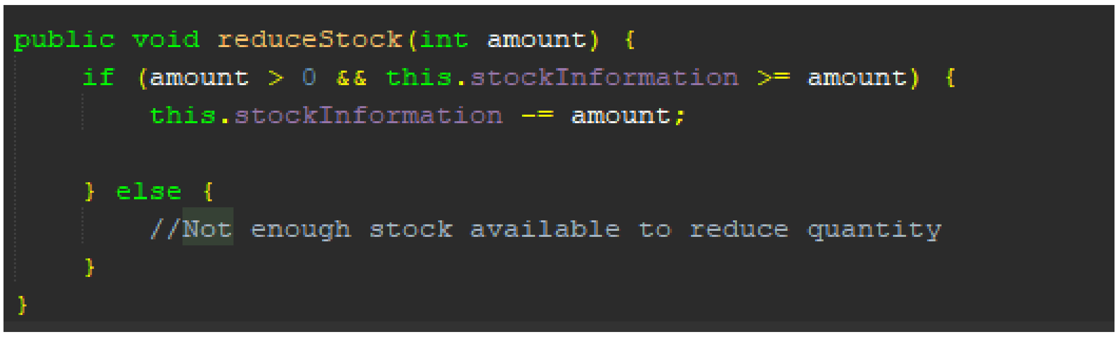
5. Getter and Setter Methods:



```java
public String getProductName() {
    return productName;
}
public void setProductName(String productName) {
    this.productName = productName;
}
public String getColor() {
    return color;
}
public void setColor(String color) {
    this.color = color;
}
public String getCategory() {
    return category;
}
public void setCategory(String category) {
    this.category = category;
}
public int getStockInformation() {
    return stockInformation;
}
public void setStockInformation(int stockInformation) {
    this.stockInformation = stockInformation;
}
public double getWeight() {
    return weight;
}
public void setWeight(double weight) {
    this.weight = weight;
}
public String getDescriptionInformation() {
    return descriptionInformation;
}
public void setDescriptionInformation(String descriptionInformation) {
    this.descriptionInformation = descriptionInformation;
}
```

These are getter and setter methods for each field. They allow access to and modification of the fields. For example, getProductName returns the value of productName, while setProductName updates the value of productName.

6. reduceStock Method:

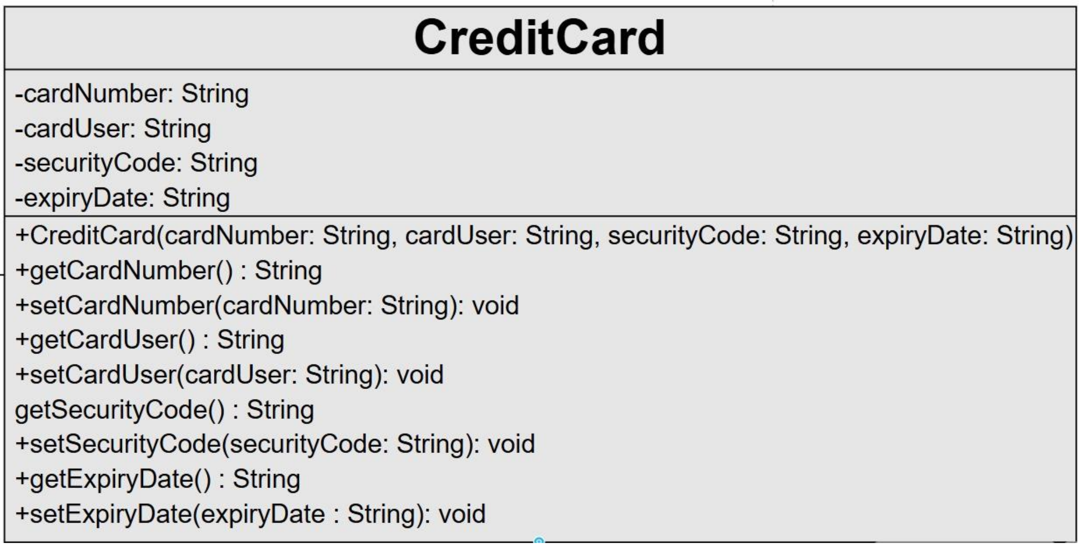


```java
public void reduceStock(int amount) {
    if (amount > 0 && this.stockInformation >= amount) {
        this.stockInformation -= amount;

    } else {
        //Not enough stock available to reduce quantity
    }
}
```

This method attempts to reduce the stock of the product by a specified amount:

- It checks if the amount is greater than 0 and if there is enough stock to reduce by that amount.
- If both conditions are met, it decreases stockInformation by the specified amount.
- If either condition is not met, it enters the else block where currently a comment indicates that there is not enough stock to reduce the quantity. This is where you might add error handling or messaging.

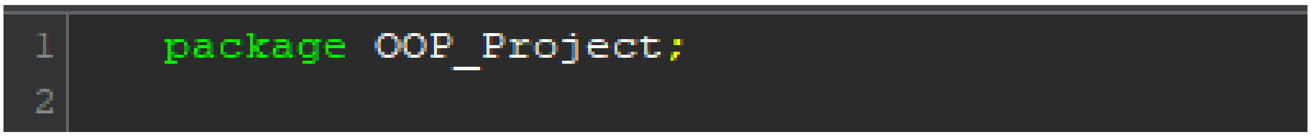
CreditCard CLASS



```
CreditCard
-cardNumber: String
-cardUser: String
-securityCode: String
-expiryDate: String
+CreditCard(cardNumber: String, cardUser: String, securityCode: String, expiryDate: String)
+getCardNumber() : String
+setCardNumber(cardNumber: String): void
+getCardUser() : String
+setCardUser(cardUser: String): void
getSecurityCode() : String
+setSecurityCode(securityCode: String): void
+getExpiryDate() : String
+setExpiryDate(expiryDate : String): void
```

The CreditCard class encapsulates the details of a credit card and provides methods to access and modify these details. The class is used to model attributes such as the credit card number, cardholder's name, security code, and expiry date.

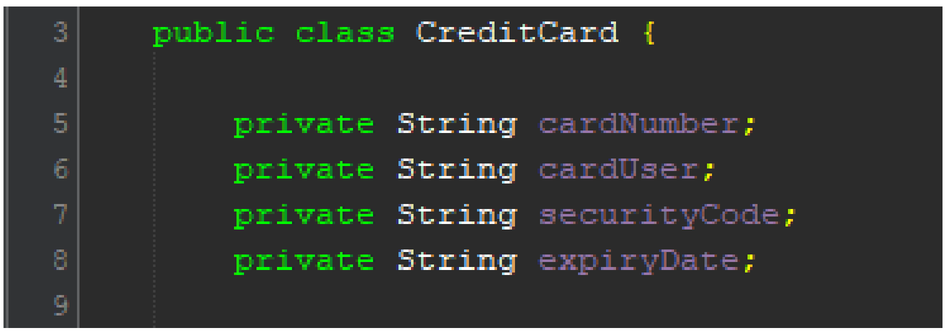
1. Package Declaration:



```java
    package OOP_Project;

```

This line specifies that the class is part of the `OOP_Project` package. In Java, packages are used to group related classes and avoid naming conflicts.

2. CreditCard Class Definition and Fields:

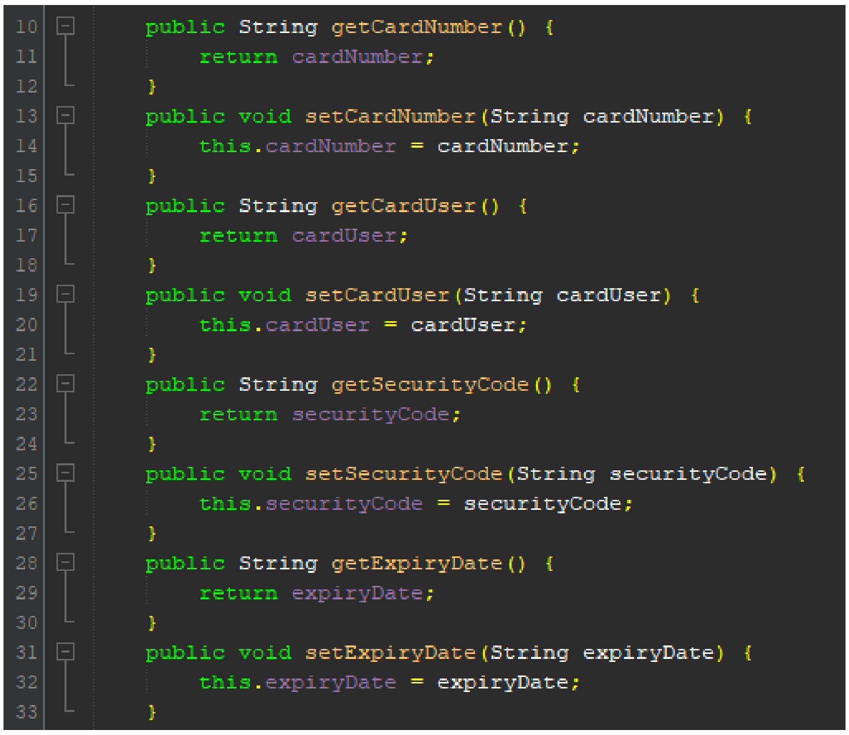


```java
public class CreditCard {

    private String cardNumber;
    private String cardUser;
    private String securityCode;
    private String expiryDate;

```

This part starts the definition of the CreditCard class and declares several private fields:

- cardNumber: The credit card number.
- cardUser: The name of the credit card holder.
- securityCode: The security code (CVV) of the credit card.
- expiryDate: The expiry date of the credit card.

3. Getter and Setter Methods:

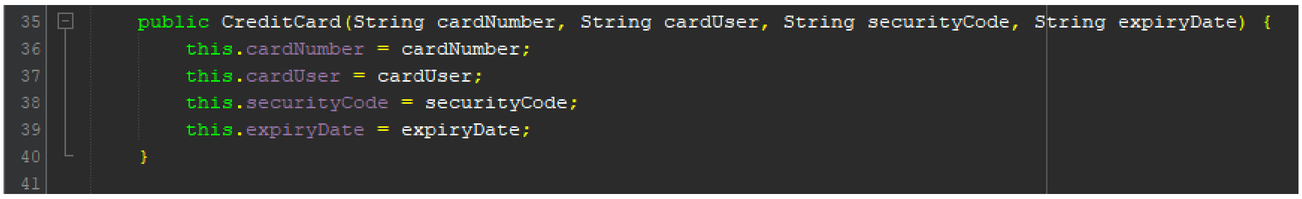


```java
    public String getCardNumber() {
        return cardNumber;
    }
    public void setCardNumber(String cardNumber) {
        this.cardNumber = cardNumber;
    }
    public String getCardUser() {
        return cardUser;
    }
    public void setCardUser(String cardUser) {
        this.cardUser = cardUser;
    }
    public String getSecurityCode() {
        return securityCode;
    }
    public void setSecurityCode(String securityCode) {
        this.securityCode = securityCode;
    }
    public String getExpiryDate() {
        return expiryDate;
    }
    public void setExpiryDate(String expiryDate) {
        this.expiryDate = expiryDate;
    }
```

These methods allow external code to access and modify the private fields:

- getCardNumber: Returns the value of the cardNumber field.
- setCardNumber: Sets the value of the cardNumber field.
- getCardUser: Returns the value of the cardUser field.
- setCardUser: Sets the value of the cardUser field.
- getSecurityCode: Returns the value of the securityCode field.
- setSecurityCode: Sets the value of the securityCode field.
- getExpiryDate: Returns the value of the expiryDate field.
- setExpiryDate: Sets the value of the expiryDate field.

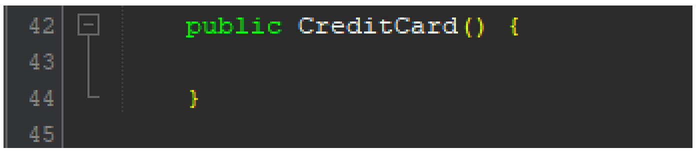
4. Constructor Methods:



```java
    public CreditCard(String cardNumber, String cardUser, String securityCode, String expiryDate) {
        this.cardNumber = cardNumber;
        this.cardUser = cardUser;
        this.securityCode = securityCode;
        this.expiryDate = expiryDate;
    }

```

This is a parameterized constructor that initializes a CreditCard object with the given values for cardNumber, cardUser, securityCode, and expiryDate. This constructor provides initial values when creating a credit card object.

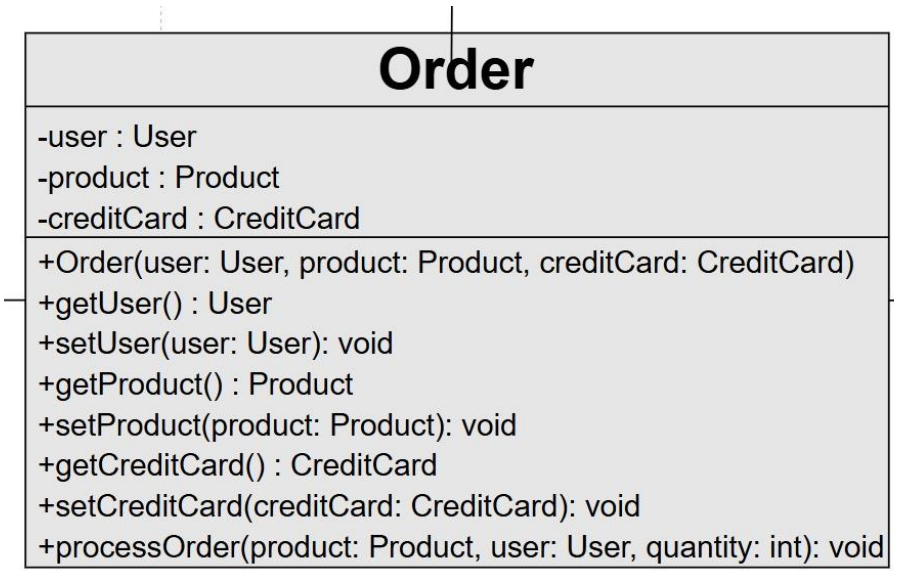
5. Default Constructor:



```java
    public CreditCard() {

    }

```

This is a no-argument constructor that does not perform any actions. It is called a default constructor and allows the creation of a CreditCard object without providing initial values for the fields.

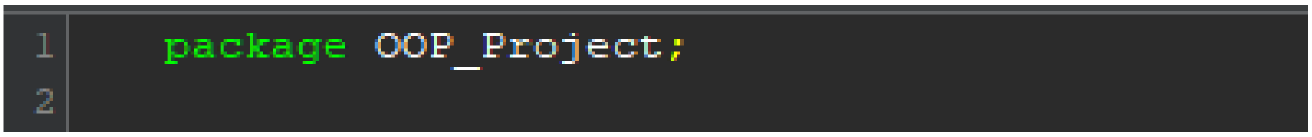
ORDER CLASS



```
Order
-user : User
-product : Product
-creditCard : CreditCard
+Order(user: User, product: Product, creditCard: CreditCard)
+getUser() : User
+setUser(user: User): void
+getProduct() : Product
+setProduct(product: Product): void
+getCreditCard() : CreditCard
+setCreditCard(creditCard: CreditCard): void
+processOrder(product: Product, user: User, quantity: int): void
```

The Order class encapsulates the details of an order, including the user placing the order, the product ordered, and the credit card used for the order. It provides methods to access and modify these details and includes a method to process the order by updating the product stock and the user's order history. Here is a step-by-step explanation of the code:

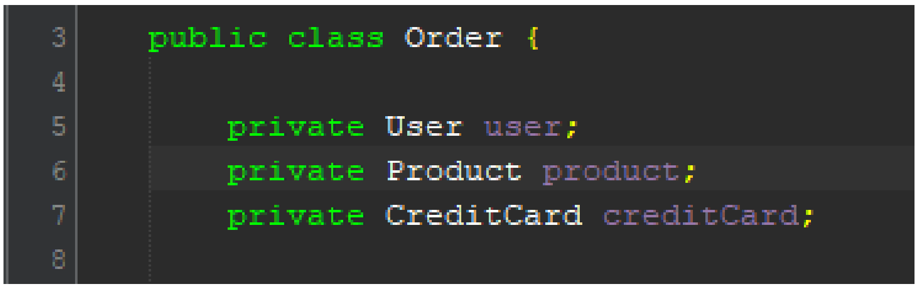
1. Package Declaration:



```java
package OOP_Project;
```

This line specifies that the class is part of the OOP_Project package. In Java, packages are used to group related classes and avoid naming conflicts.

2. Order Class Definition and Fields:

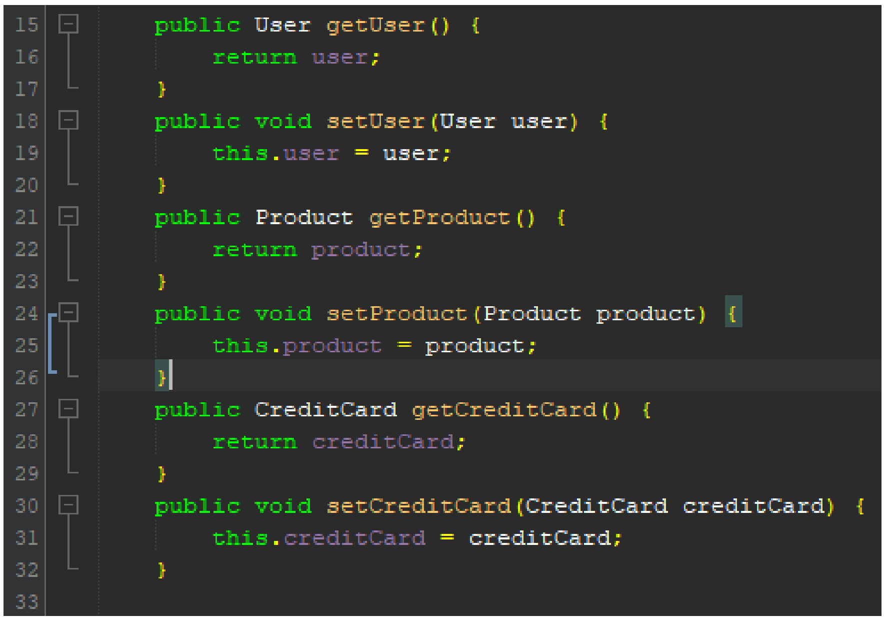


```java
public class Order {

    private User user;
    private Product product;
    private CreditCard creditCard;

```

This part starts the definition of the Order class and declares several private fields:

- user: The user who placed the order.
- product: The product being ordered.
- creditCard: The credit card used for the order.

3. Getter and Setter Methods:

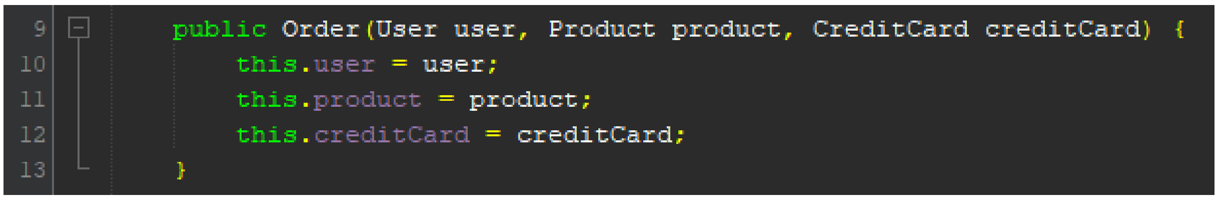


```java
    public User getUser() {
        return user;
    }
    public void setUser(User user) {
        this.user = user;
    }
    public Product getProduct() {
        return product;
    }
    public void setProduct(Product product) {
        this.product = product;
    }
    public CreditCard getCreditCard() {
        return creditCard;
    }
    public void setCreditCard(CreditCard creditCard) {
        this.creditCard = creditCard;
    }
```

These methods allow external code to access and modify the private fields:

- getUser: Returns the value of the user field.
- setUser: Sets the value of the user field.
- getProduct: Returns the value of the product field.
- setProduct: Sets the value of the product field.
- getCreditCard: Returns the value of the creditCard field.
- setCreditCard: Sets the value of the creditCard field.

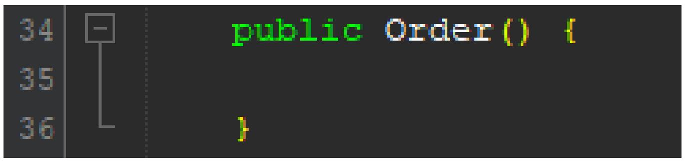
4. Constructor Methods:



```java
    public Order(User user, Product product, CreditCard creditCard) {
        this.user = user;
        this.product = product;
        this.creditCard = creditCard;
    }
```

This is a parameterized constructor that initializes an Order object with the given user, product, and creditCard. This constructor provides initial values when creating an order object.

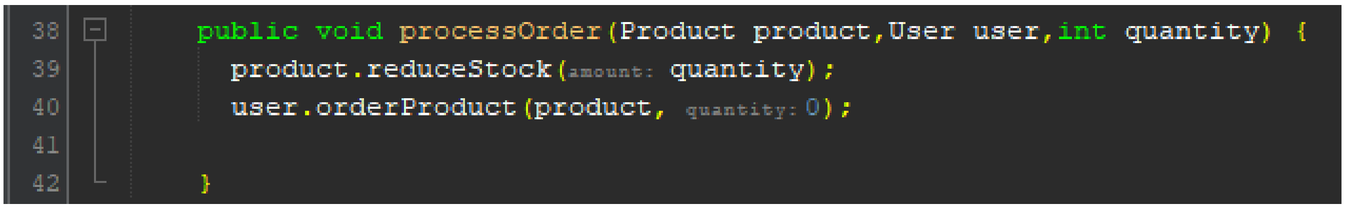
5. Default Constructor:



```java
    public Order() {

    }
```

This is a no-argument constructor that does not perform any actions. It is called a default constructor and allows the creation of an Order object without providing initial values for the fields.

6. processOrder Method:

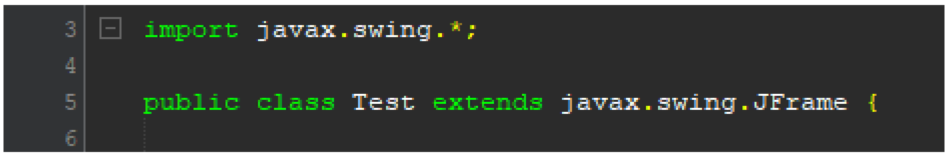


```java
    public void processOrder(Product product,User user,int quantity) {
      product.reduceStock(amount: quantity);
      user.orderProduct(product, quantity: 0);

    }
```

This method processes an order by:

- Reducing the stock of the product by the specified quantity using the reduceStock method of the Product class.
- Adding the product to the user's list of ordered products using the orderProduct method of the User class. Note that the second parameter of orderProduct is set to 0, which might need to be adjusted based on the actual requirements of that method.

TEST CLASS

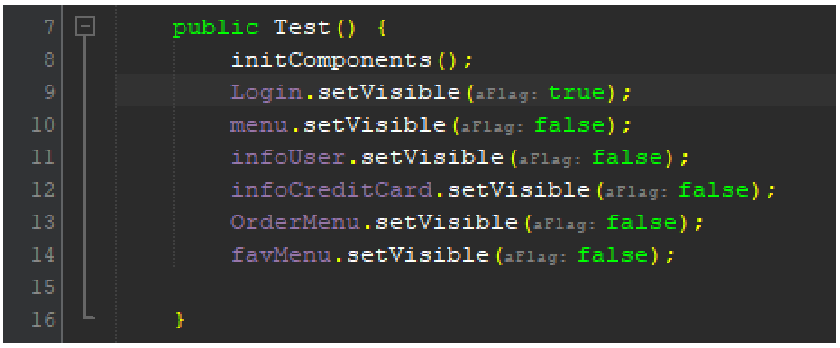


```java
import javax.swing.*;

public class Test extends javax.swing.JFrame {

```

Test Class, the main application class and creates the user interface.

Java Swing library, used to create the interface.

Constructor Methods:



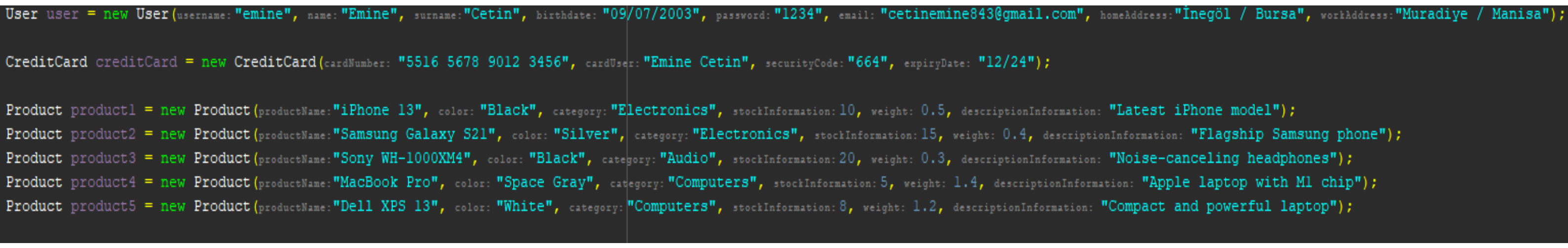
```java
    public Test() {
        initComponents();
        Login.setVisible(aFlag: true);
        menu.setVisible(aFlag: false);
        infoUser.setVisible(aFlag: false);
        infoCreditCard.setVisible(aFlag: false);
        OrderMenu.setVisible(aFlag: false);
        favMenu.setVisible(aFlag: false);

    }
```

This Test() constructor method initializes and configures the user interface components created using the Java Swing library. It subsequently hides these components.

Initialized Components:

- menu: Represents the main menu component.
- infoUser: A panel component displaying user information.
- infoCreditCard: A panel component displaying credit card information.
- OrderMenu: A panel component showing the order menu.
- favMenu: A panel component displaying favorite products.
- Login: A panel component that displays login information.

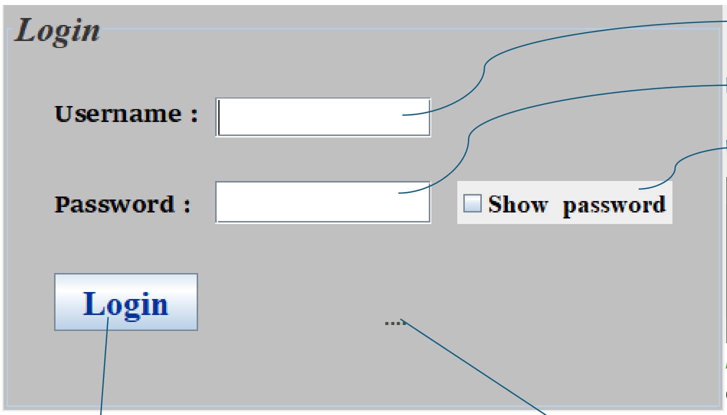
Creating Objects:



```java
User user = new User(username: "emine", name: "Emine", surname: "Cetin", birthdate: "09/07/2003", password: "1234", email: "cetinemine843@gmail.com", homeAddress: "İnegöl / Bursa", workAddress: "Muradiye / Manisa");

CreditCard creditCard = new CreditCard(cardNumber: "5516 5678 9012 3456", cardUser: "Emine Cetin", securityCode: "664", expiryDate: "12/24");

Product product1 = new Product(productName: "iPhone 13", color: "Black", category: "Electronics", stockInformation: 10, weight: 0.5, descriptionInformation: "Latest iPhone model");
Product product2 = new Product(productName: "Samsung Galaxy S21", color: "Silver", category: "Electronics", stockInformation: 15, weight: 0.4, descriptionInformation: "Flagship Samsung phone");
Product product3 = new Product(productName: "Sony WH-1000XM4", color: "Black", category: "Audio", stockInformation: 20, weight: 0.3, descriptionInformation: "Noise-canceling headphones");
Product product4 = new Product(productName: "MacBook Pro", color: "Space Gray", category: "Computers", stockInformation: 5, weight: 1.4, descriptionInformation: "Apple laptop with M1 chip");
Product product5 = new Product(productName: "Dell XPS 13", color: "White", category: "Computers", stockInformation: 8, weight: 1.2, descriptionInformation: "Compact and powerful laptop");
```

We created user information, product information and credit card information using User, CreditCard and Product classes.

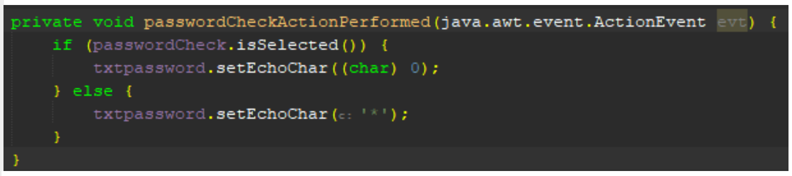
Login Panel:



txtusername: Enter a username.

txtpassword: Enter a password.

passwordCheck:

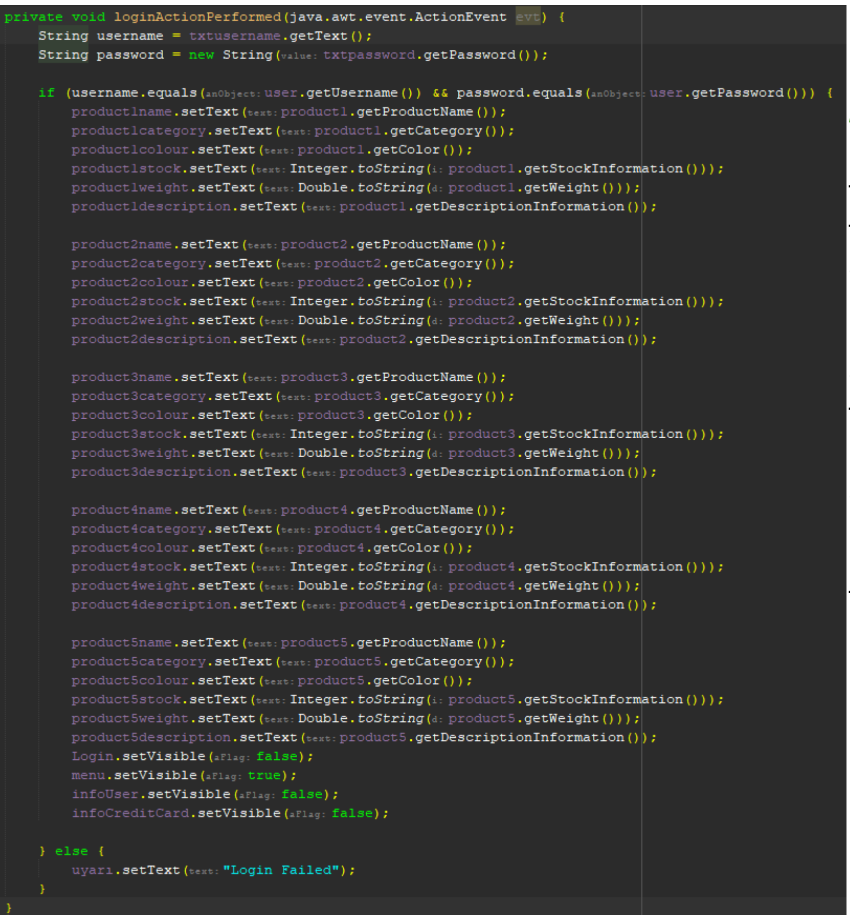


```java
private void passwordCheckActionPerformed(java.awt.event.ActionEvent evt) {
    if (passwordCheck.isSelected()) {
        txtpassword.setEchoChar((char) 0);
    } else {
        txtpassword.setEchoChar(c: '*');
    }
}
```

Figure 1When selected, this checkbox changes the visibility of characters entered in the text field named txtpassword. If the checkbox is selected, the characters are visible (the “echo” character is set to 0) and if not selected, the characters are hidden (the “echo character is set to '*').

Login:

uyarı: If the login fails, the text 'Login failed' will show.



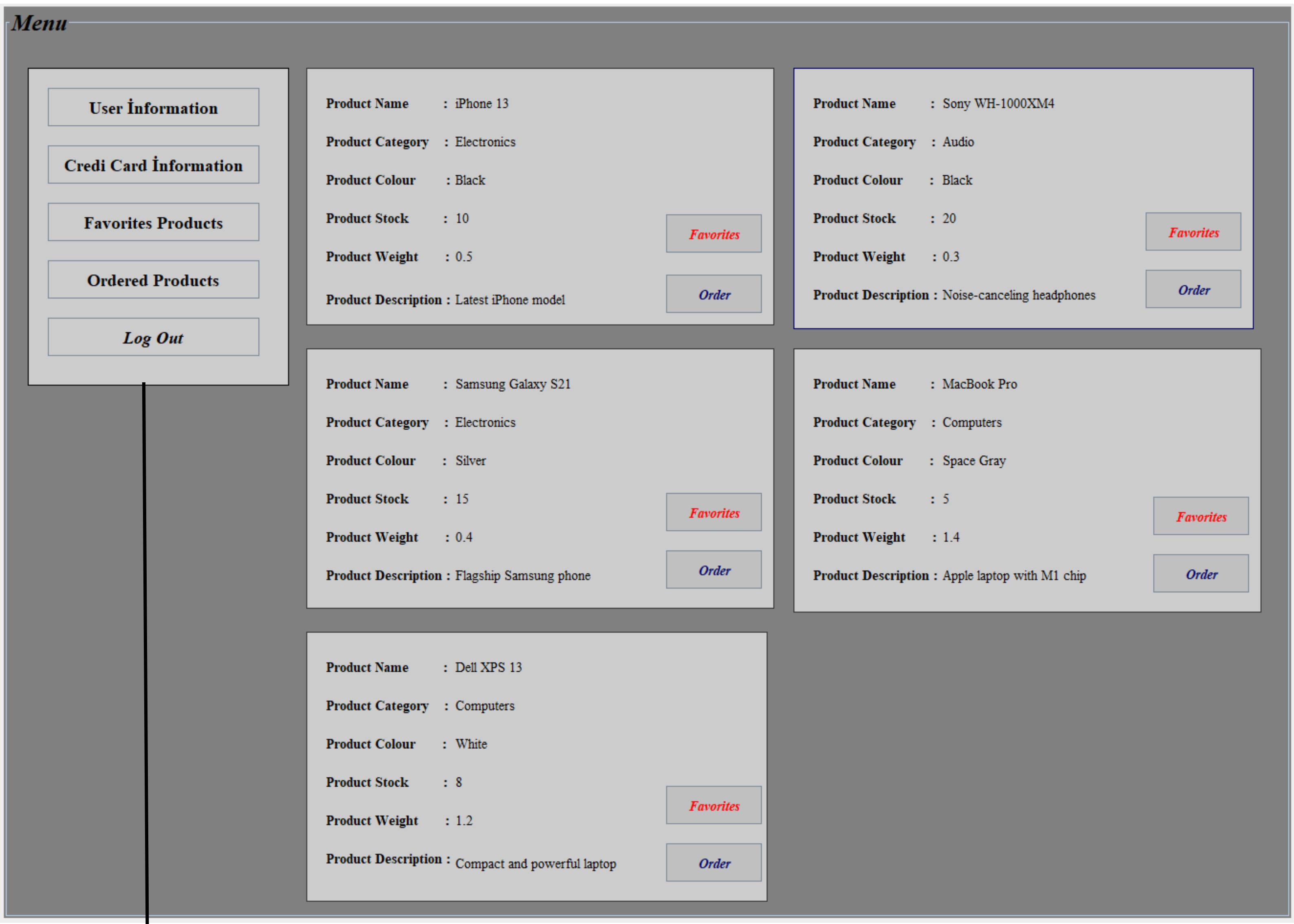
```java
private void loginActionPerformed(java.awt.event.ActionEvent evt) {
    String username = txtusername.getText();
    String password = new String(value: txtpassword.getPassword());

    if (username.equals(anObject: user.getUsername()) && password.equals(anObject: user.getPassword())) {
        product1name.setText(text: product1.getProductName());
        product1category.setText(text: product1.getCategory());
        product1colour.setText(text: product1.getColor());
        product1stock.setText(text: Integer.toString(i: product1.getStockInformation()));
        product1weight.setText(text: Double.toString(d: product1.getWeight()));
        product1description.setText(text: product1.getDescriptionInformation());

        product2name.setText(text: product2.getProductName());
        product2category.setText(text: product2.getCategory());
        product2colour.setText(text: product2.getColor());
        product2stock.setText(text: Integer.toString(i: product2.getStockInformation()));
        product2weight.setText(text: Double.toString(d: product2.getWeight()));
        product2description.setText(text: product2.getDescriptionInformation());

        product3name.setText(text: product3.getProductName());
        product3category.setText(text: product3.getCategory());
        product3colour.setText(text: product3.getColor());
        product3stock.setText(text: Integer.toString(i: product3.getStockInformation()));
        product3weight.setText(text: Double.toString(d: product3.getWeight()));
        product3description.setText(text: product3.getDescriptionInformation());

        product4name.setText(text: product4.getProductName());
        product4category.setText(text: product4.getCategory());
        product4colour.setText(text: product4.getColor());
        product4stock.setText(text: Integer.toString(i: product4.getStockInformation()));
        product4weight.setText(text: Double.toString(d: product4.getWeight()));
        product4description.setText(text: product4.getDescriptionInformation());

        product5name.setText(text: product5.getProductName());
        product5category.setText(text: product5.getCategory());
        product5colour.setText(text: product5.getColor());
        product5stock.setText(text: Integer.toString(i: product5.getStockInformation()));
        product5weight.setText(text: Double.toString(d: product5.getWeight()));
        product5description.setText(text: product5.getDescriptionInformation());
        Login.setVisible(aFlag: false);
        menu.setVisible(aFlag: true);
        infoUser.setVisible(aFlag: false);
        infoCreditCard.setVisible(aFlag: false);

    } else {
        uyarı.setText(text: "Login Failed");
    }
}
```

Figure 2This button takes the entered username and password from the text fields (txtusername and txtpassword respectively), then checks if they match the username and password stored in the User object.

If the login credentials match, it populates labels with objects created from the Product class. It then hides the login interface (Login), shows a menu interface (menu) and hides some other components (infoUser and infoCreditCard).

If the login fails, it sets a label (uyarı) text to indicate that the login failed.

Menu Panel:



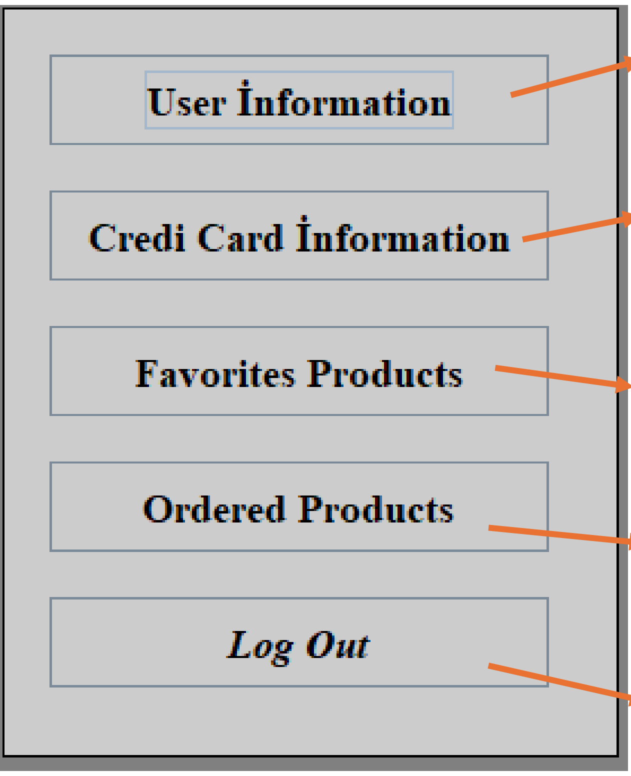
Product Name : iPhone 13
Product Category : Electronics
Product Colour : Black
Product Stock : 10
Product Weight : 0.5
Product Description : Latest iPhone model

Product Name : Sony WH-1000XM4
Product Category : Audio
Product Colour : Black
Product Stock : 20
Product Weight : 0.3
Product Description : Noise-canceling headphones

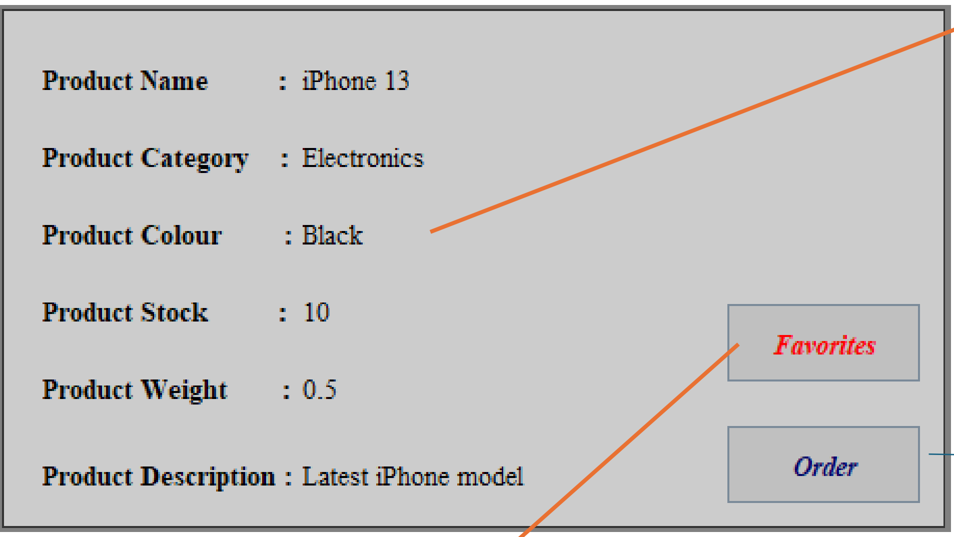
Product Name : Samsung Galaxy S21
Product Category : Electronics
Product Colour : Silver
Product Stock : 15
Product Weight : 0.4
Product Description : Flagship Samsung phone

Product Name : MacBook Pro
Product Category : Computers
Product Colour : Space Gray
Product Stock : 5
Product Weight : 1.4
Product Description : Apple laptop with M1 chip

Product Name : Dell XPS 13
Product Category : Computers
Product Colour : White
Product Stock : 8
Product Weight : 1.2
Product Description : Compact and powerful laptop



- Opens the “infoUser” panel showing user information.
- Opens the “infoCredirCard” panel showing credit card information.
- Opens the “favMenu” panel showing the products added to the favorites.
- Opens the “OrderMenu” panel showing the ordered products.
- Makes the user log out.



Product Name : iPhone 13

Product Category : Electronics

Product Colour : Black

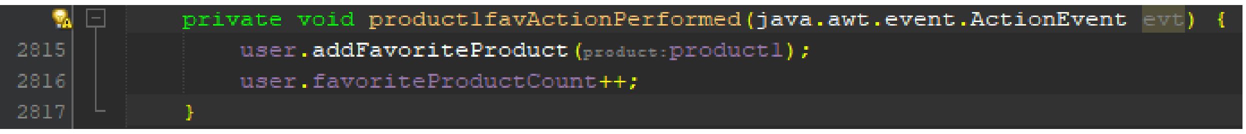
Product Stock : 10

Product Weight : 0.5

Product Description : Latest iPhone model

The properties of the product1 object we created in the product class are written to the labels in the "jProduct1" panel.

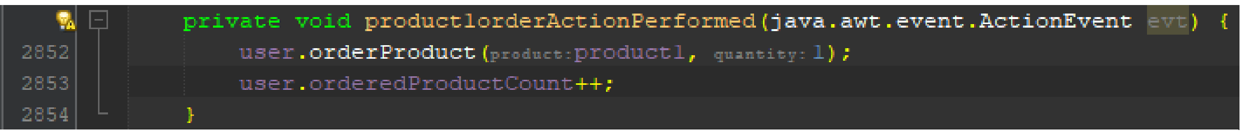
product1fav:



```
private void product1favActionPerformed(java.awt.event.ActionEvent evt) {
    user.addFavoriteProduct(product:product1);
    user.favoriteProductCount++;
}
```

Figure 3 When the "product1fav" button is clicked, product1 is added to the user's favorite products list and favoriteProductCount is increased by one for the number of favorite products.

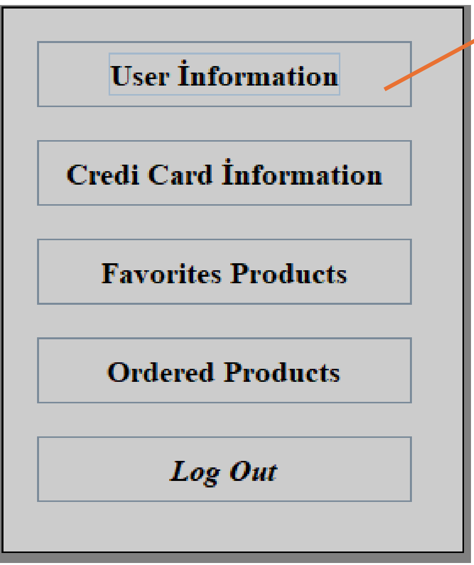
product1order:



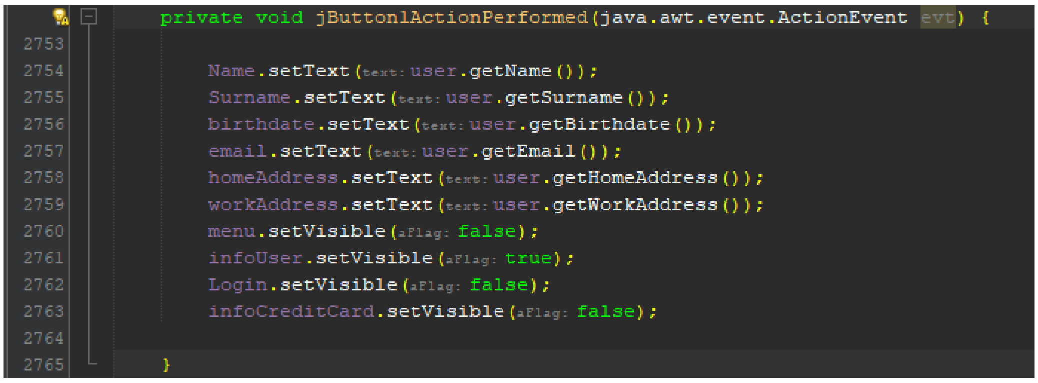
```
private void product1orderActionPerformed(java.awt.event.ActionEvent evt) {
    user.orderProduct(product:product1, quantity: 1);
    user.orderedProductCount++;
}
```

Figure 4 When the "product1order" button is clicked, product1 is added to the user's ordered products list and the orderedProductCount is increased by one for the number of ordered products.

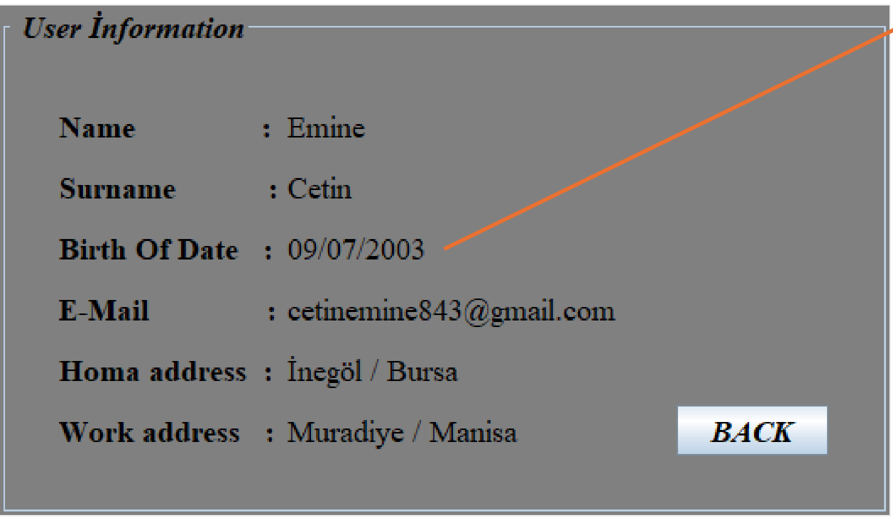
infoUser Panel:



When the button is clicked, user information is written to the labels.



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    Name.setText(text: user.getName());
    Surname.setText(text: user.getSurname());
    birthdate.setText(text: user.getBirthdate());
    email.setText(text: user.getEmail());
    homeAddress.setText(text: user.getHomeAddress());
    workAddress.setText(text: user.getWorkAddress());
    menu.setVisible(aFlag: false);
    infoUser.setVisible(aFlag: true);
    Login.setVisible(aFlag: false);
    infoCreditCard.setVisible(aFlag: false);

}
```

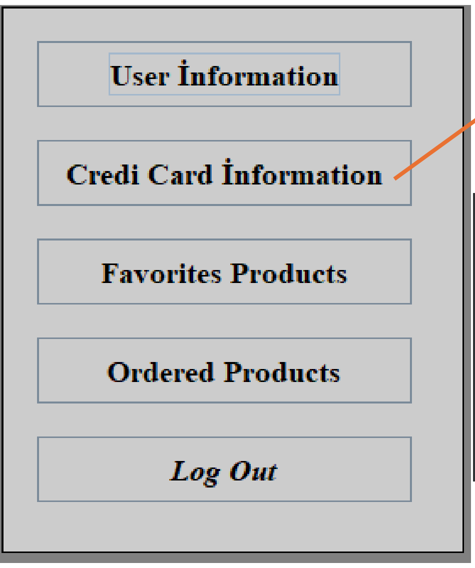


User İnformation

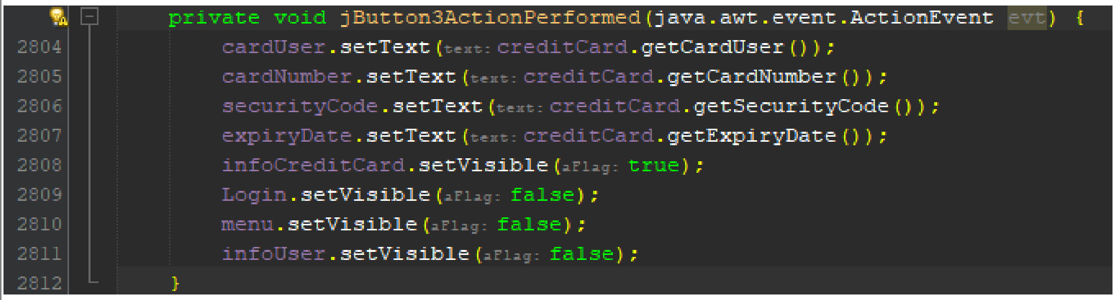
Name	:	Emine
Surname	:	Cetin
Birth Of Date	:	09/07/2003
E-Mail	:	cetinemine843@gmail.com
Homa address	:	İnegöl / Bursa
Work address	:	Muradiye / Manisa

The properties of the user object we created in the user class are written to the labels in the "infoUser" panel.

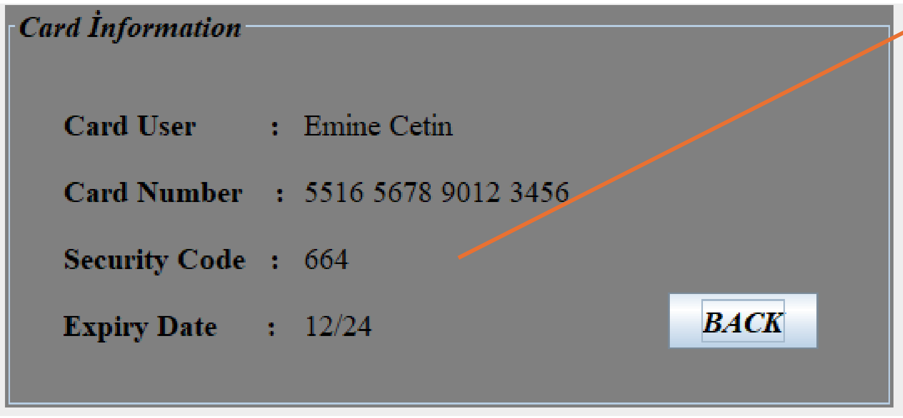
infoCreditCard Panel:



When the button is clicked, credit card information is written to the labels.



```
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        cardUser.setText(text: creditCard.getCardUser());
        cardNumber.setText(text: creditCard.getCardNumber());
        securityCode.setText(text: creditCard.getSecurityCode());
        expiryDate.setText(text: creditCard.getExpiryDate());
        infoCreditCard.setVisible(aFlag: true);
        Login.setVisible(aFlag: false);
        menu.setVisible(aFlag: false);
        infoUser.setVisible(aFlag: false);
    }
```

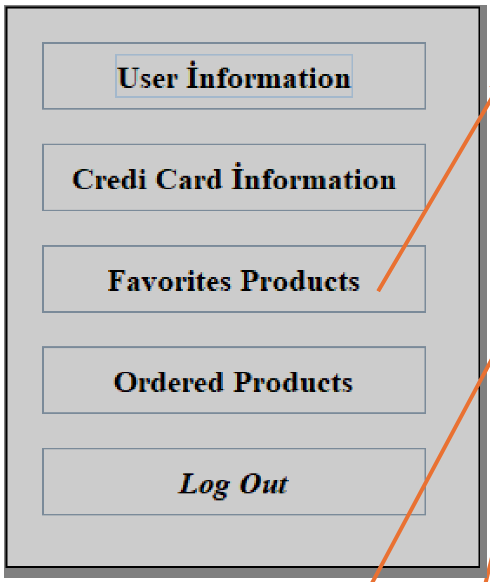


Card İnformation

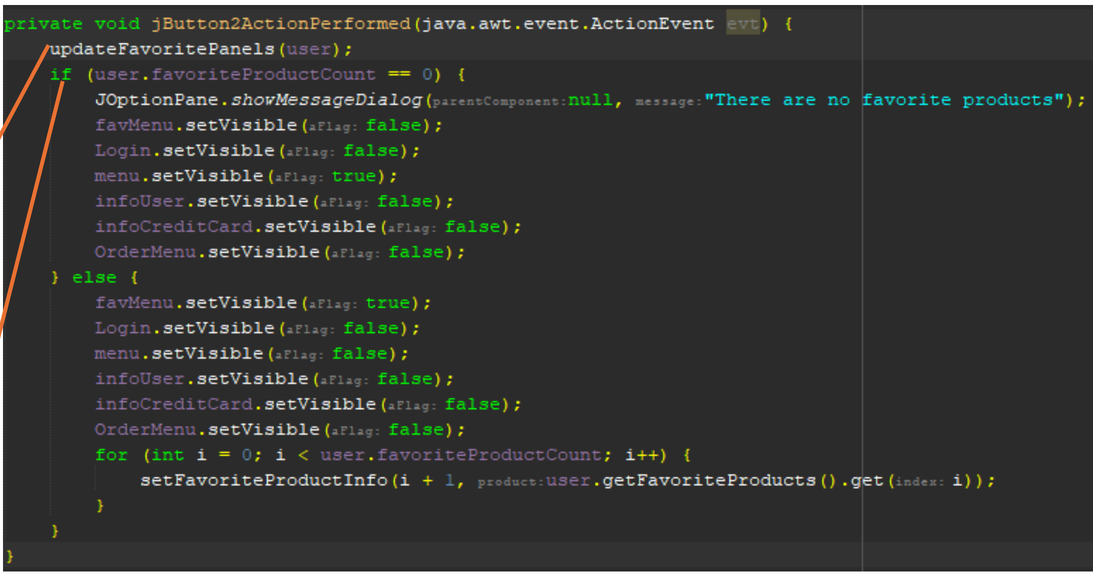
Card User	:	Emine Cetin
Card Number	:	5516 5678 9012 3456
Security Code	:	664
Expiry Date	:	12/24

The properties of the creditCard object we created in the CreditCard class are written to the labels in the “infoCreditCard” panel.

favMenu Panel:

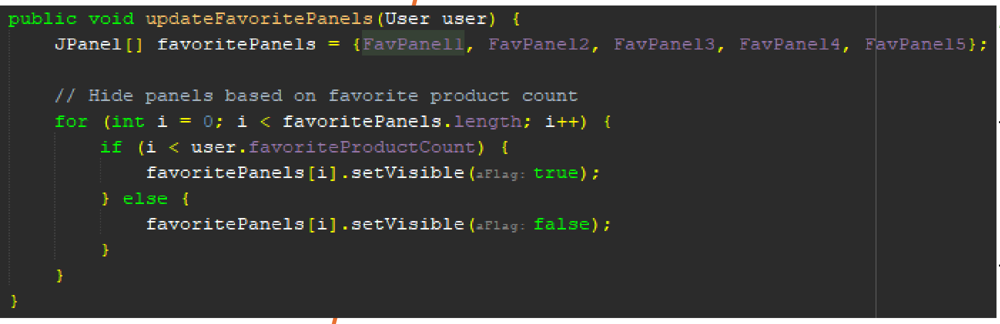


When the button is clicked, update the user's favorite products, and show or hide the appropriate panels in the user interface.



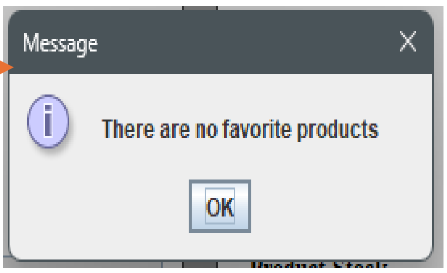
```java
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    updateFavoritePanels(user);
    if (user.favoriteProductCount == 0) {
        JOptionPane.showMessageDialog(parentComponent:null, message:"There are no favorite products");
        favMenu.setVisible(aFlag: false);
        Login.setVisible(aFlag: false);
        menu.setVisible(aFlag: true);
        infoUser.setVisible(aFlag: false);
        infoCreditCard.setVisible(aFlag: false);
        OrderMenu.setVisible(aFlag: false);
    } else {
        favMenu.setVisible(aFlag: true);
        Login.setVisible(aFlag: false);
        menu.setVisible(aFlag: false);
        infoUser.setVisible(aFlag: false);
        infoCreditCard.setVisible(aFlag: false);
        OrderMenu.setVisible(aFlag: false);
        for (int i = 0; i < user.favoriteProductCount; i++) {
            setFavoriteProductInfo(i + 1, product:user.getFavoriteProducts().get(index: i));
        }
    }
}
```

- **updateFavoritePanels(user);** This line calls the updateFavoritePanels method to update the user's favorite product panels.



```java
public void updateFavoritePanels(User user) {
    JPanel[] favoritePanels = {FavPanel1, FavPanel2, FavPanel3, FavPanel4, FavPanel5};

    // Hide panels based on favorite product count
    for (int i = 0; i < favoritePanels.length; i++) {
        if (i < user.favoriteProductCount) {
            favoritePanels[i].setVisible(aFlag: true);
        } else {
            favoritePanels[i].setVisible(aFlag: false);
        }
    }
}
```

Figure 5 Assigns five favorite panels to an array. The loop adjusts the visibility of each panel according to the number of the user's favorite products. The loop checks each panel in the favoritePanels array. If the panel's index (i) is less than the user's favorite product count (user.favoriteProductCount), the panel is made visible. Otherwise, the panel is hidden.



There are no favorite products

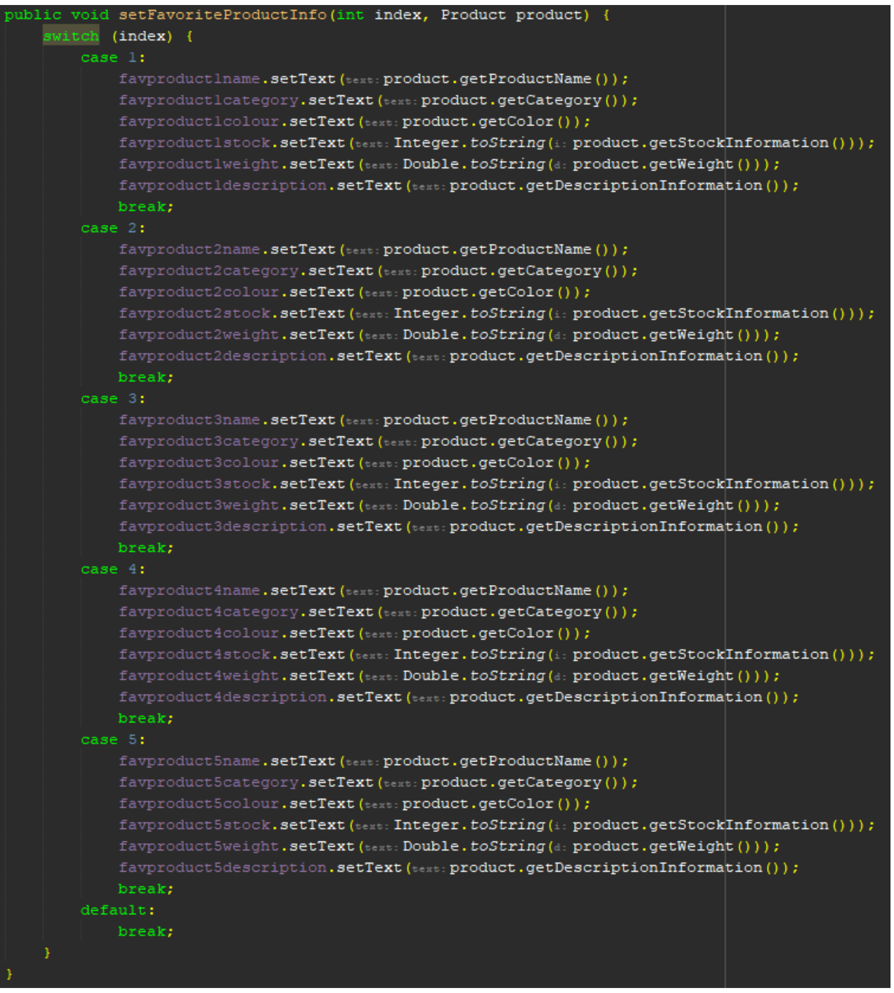
- **if (user.favoriteProductCount == 0) {**

This condition checks the number of the user's favorite products.

→If there are no favorite products, a message box will pop up and show the user the message "There are no favorite products". Then, the favorite menu (favMenu) and other related panels (Login, infoUser, infoCreditCard, OrderMenu) are hidden and the main menu is made visible.

→If there is a favorite product, the favorite menu is made visible and other panels are hidden. Then, a loop updates the information of the user's favorite products. This is done using the "setFavoriteProductInfo" method and is performed sequentially for each favorite product.

- **setFavoriteProductInfo(index, product) {**



```java
public void setFavoriteProductInfo(int index, Product product) {
    switch (index) {
        case 1:
            favproduct1name.setText(text: product.getProductName());
            favproduct1category.setText(text: product.getCategory());
            favproduct1colour.setText(text: product.getColor());
            favproduct1stock.setText(text: Integer.toString(i: product.getStockInformation()));
            favproduct1weight.setText(text: Double.toString(d: product.getWeight()));
            favproduct1description.setText(text: product.getDescriptionInformation());
            break;
        case 2:
            favproduct2name.setText(text: product.getProductName());
            favproduct2category.setText(text: product.getCategory());
            favproduct2colour.setText(text: product.getColor());
            favproduct2stock.setText(text: Integer.toString(i: product.getStockInformation()));
            favproduct2weight.setText(text: Double.toString(d: product.getWeight()));
            favproduct2description.setText(text: product.getDescriptionInformation());
            break;
        case 3:
            favproduct3name.setText(text: product.getProductName());
            favproduct3category.setText(text: product.getCategory());
            favproduct3colour.setText(text: product.getColor());
            favproduct3stock.setText(text: Integer.toString(i: product.getStockInformation()));
            favproduct3weight.setText(text: Double.toString(d: product.getWeight()));
            favproduct3description.setText(text: product.getDescriptionInformation());
            break;
        case 4:
            favproduct4name.setText(text: product.getProductName());
            favproduct4category.setText(text: product.getCategory());
            favproduct4colour.setText(text: product.getColor());
            favproduct4stock.setText(text: Integer.toString(i: product.getStockInformation()));
            favproduct4weight.setText(text: Double.toString(d: product.getWeight()));
            favproduct4description.setText(text: product.getDescriptionInformation());
            break;
        case 5:
            favproduct5name.setText(text: product.getProductName());
            favproduct5category.setText(text: product.getCategory());
            favproduct5colour.setText(text: product.getColor());
            favproduct5stock.setText(text: Integer.toString(i: product.getStockInformation()));
            favproduct5weight.setText(text: Double.toString(d: product.getWeight()));
            favproduct5description.setText(text: product.getDescriptionInformation());
            break;
        default:
            break;
    }
}
```

Figure 6

The method sets the product information to the relevant components in the user interface.

index: Determines which favorite product will be updated.

product: Product object containing the information of the favorite product to update.

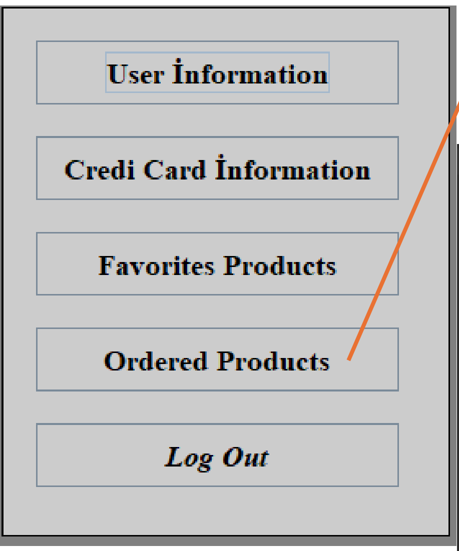
The switch structure is used to perform operations corresponding to a specific index.

Each case block updates the texts of the components of a specific favorite product.

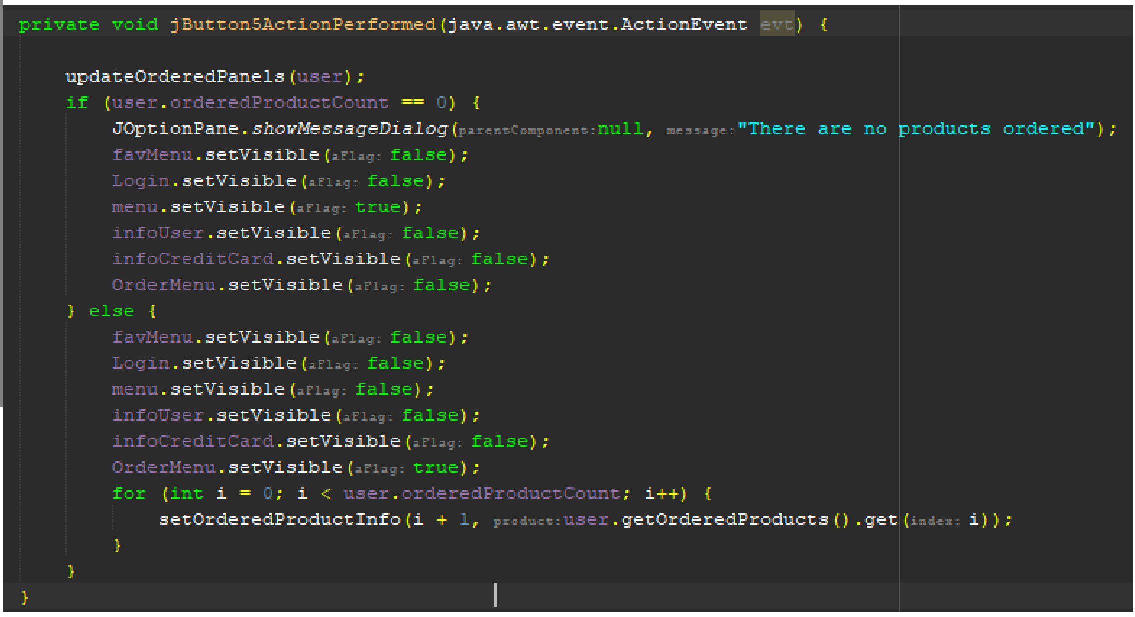
These operations are repeated from case 1 to 5, each representing a specific favorite product.

The default block does not take any action when the index parameter is not in the range 1-5.

OrderMenu Panel:

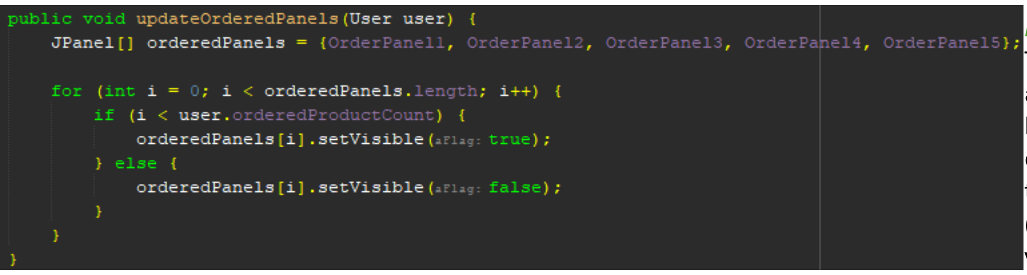


When the button is clicked, update the user's ordered products, and show or hide the appropriate panels in the user interface.



```java
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

    updateOrderedPanels(user);
    if (user.orderedProductCount == 0) {
        JOptionPane.showMessageDialog(parentComponent:null, message:"There are no products ordered");
        favMenu.setVisible(aFlag: false);
        Login.setVisible(aFlag: false);
        menu.setVisible(aFlag: true);
        infoUser.setVisible(aFlag: false);
        infoCreditCard.setVisible(aFlag: false);
        OrderMenu.setVisible(aFlag: false);
    } else {
        favMenu.setVisible(aFlag: false);
        Login.setVisible(aFlag: false);
        menu.setVisible(aFlag: false);
        infoUser.setVisible(aFlag: false);
        infoCreditCard.setVisible(aFlag: false);
        OrderMenu.setVisible(aFlag: true);
        for (int i = 0; i < user.orderedProductCount; i++) {
            setOrderedProductInfo(i + 1, product:user.getOrderedProducts().get(index: i));
        }
    }
}
```

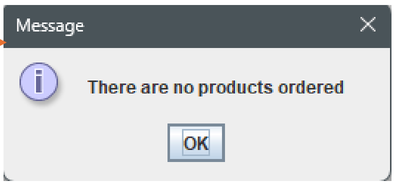
- **updateOrderedPanels(user);** This line calls the updateOrderedPanels method to update the user's order product panels.



```
public void updateOrderedPanels(User user) {
    JPanel[] orderedPanels = {OrderPanel1, OrderPanel2, OrderPanel3, OrderPanel4, OrderPanel5};

    for (int i = 0; i < orderedPanels.length; i++) {
        if (i < user.orderedProductCount) {
            orderedPanels[i].setVisible(aFlag: true);
        } else {
            orderedPanels[i].setVisible(aFlag: false);
        }
    }
}
```

Figure 7 Assigns five ordered panels to an array. The loop adjusts the visibility of each panel according to the number of products the user has ordered. The loop checks each panel in the orderedPanels array. If the panel's index (i) is less than the number of products ordered by the user (user.orderedProductCount), the panel is made visible. Otherwise, the panel is hidden.

- **if (user.orderedProductCount == 0) {**



Message

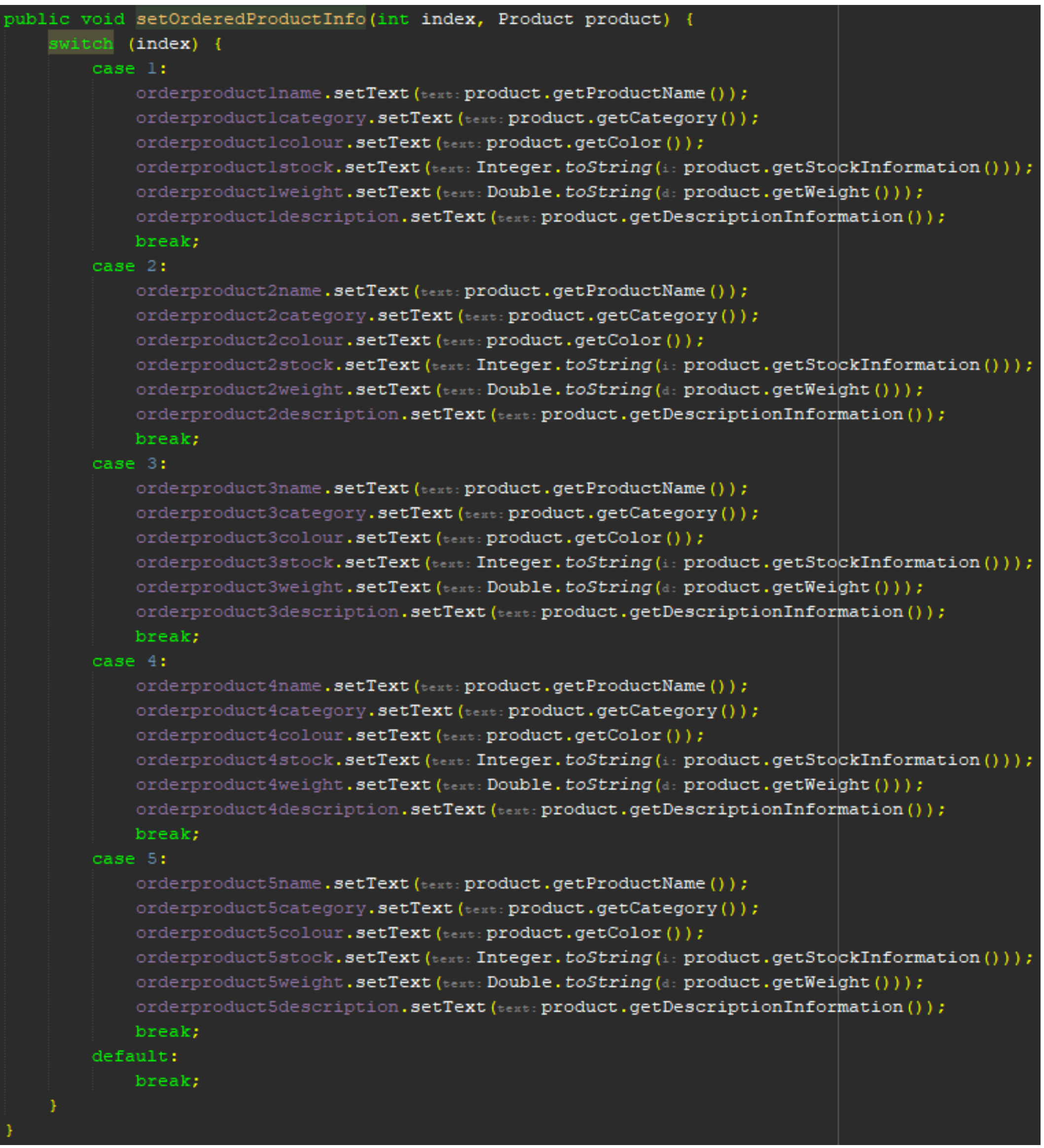
There are no products ordered

This condition controls the number of the user's ordered products.

→If there are no products ordered, a message box will pop up and the user will be shown the message "There are no products ordered". Then, the favorite menu (favMenu), login screen (Login), user info screen (infoUser), credit card info screen (infoCreditCard) and order menu (OrderMenu) are hidden, and the main menu is made visible.

→If there is an ordered product, the favorite menu, login screen, main menu, user info screen and credit card info screen are hidden, and the order menu is made visible. Then, in a loop, the information of the products ordered by the user is updated. This operation is done using the "setOrderedProductInfo" method and is performed sequentially for each ordered product.

setOrderedProductInfo(index, product) {



```
public void setOrderedProductInfo(int index, Product product) {
    switch (index) {
        case 1:
            orderproduct1name.setText(text: product.getProductName());
            orderproduct1category.setText(text: product.getCategory());
            orderproduct1colour.setText(text: product.getColor());
            orderproduct1stock.setText(text: Integer.toString(i: product.getStockInformation()));
            orderproduct1weight.setText(text: Double.toString(d: product.getWeight()));
            orderproduct1description.setText(text: product.getDescriptionInformation());
            break;
        case 2:
            orderproduct2name.setText(text: product.getProductName());
            orderproduct2category.setText(text: product.getCategory());
            orderproduct2colour.setText(text: product.getColor());
            orderproduct2stock.setText(text: Integer.toString(i: product.getStockInformation()));
            orderproduct2weight.setText(text: Double.toString(d: product.getWeight()));
            orderproduct2description.setText(text: product.getDescriptionInformation());
            break;
        case 3:
            orderproduct3name.setText(text: product.getProductName());
            orderproduct3category.setText(text: product.getCategory());
            orderproduct3colour.setText(text: product.getColor());
            orderproduct3stock.setText(text: Integer.toString(i: product.getStockInformation()));
            orderproduct3weight.setText(text: Double.toString(d: product.getWeight()));
            orderproduct3description.setText(text: product.getDescriptionInformation());
            break;
        case 4:
            orderproduct4name.setText(text: product.getProductName());
            orderproduct4category.setText(text: product.getCategory());
            orderproduct4colour.setText(text: product.getColor());
            orderproduct4stock.setText(text: Integer.toString(i: product.getStockInformation()));
            orderproduct4weight.setText(text: Double.toString(d: product.getWeight()));
            orderproduct4description.setText(text: product.getDescriptionInformation());
            break;
        case 5:
            orderproduct5name.setText(text: product.getProductName());
            orderproduct5category.setText(text: product.getCategory());
            orderproduct5colour.setText(text: product.getColor());
            orderproduct5stock.setText(text: Integer.toString(i: product.getStockInformation()));
            orderproduct5weight.setText(text: Double.toString(d: product.getWeight()));
            orderproduct5description.setText(text: product.getDescriptionInformation());
            break;
        default:
            break;
    }
}
```

Figure 8

The method sets the product information to the relevant components in the user interface.

index: Determines which ordered product will be updated.

product: Product object containing the information of the ordered product to update.

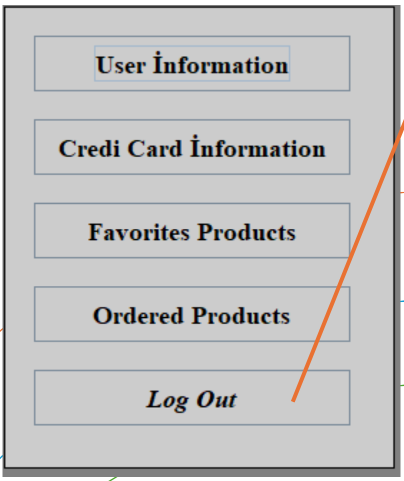
The switch structure is used to perform operations corresponding to a specific index.

Each case block updates the texts of the components of a specific ordered product.

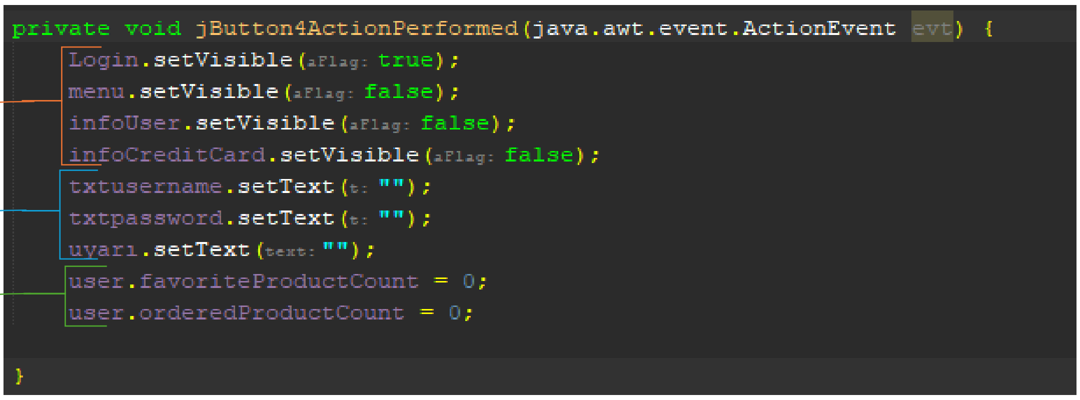
These operations are repeated from case 1 to 5, each representing a specific ordered product.

The default block does not take any action when the index parameter is not in the range 1-5.

Log Out:



When the button is clicked, the user out of the account.



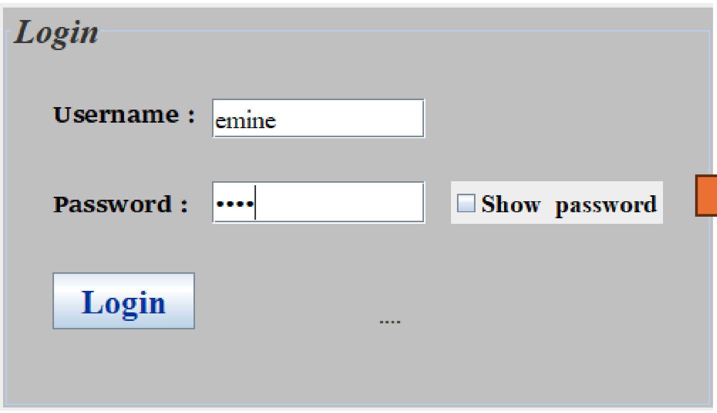
```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    Login.setVisible(aFlag: true);
    menu.setVisible(aFlag: false);
    infoUser.setVisible(aFlag: false);
    infoCreditCard.setVisible(aFlag: false);
    txtusername.setText(t: "");
    txtpassword.setText(t: "");
    uyari.setText(text: "");
    user.favoriteProductCount = 0;
    user.orderedProductCount = 0;

}
```

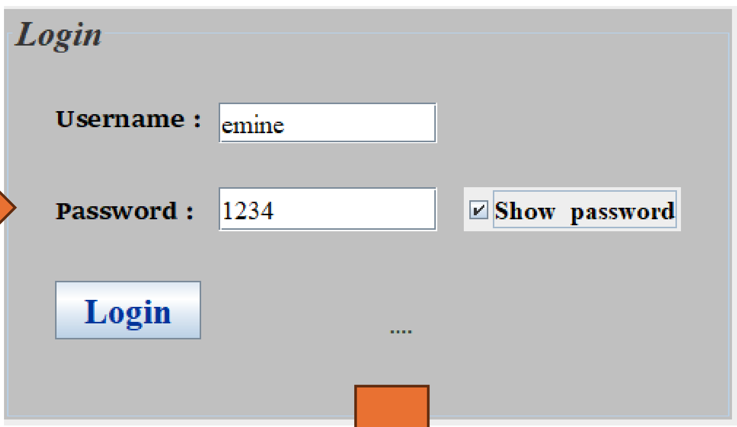
These lines hide the main menu (menu), user information screen (infoUser) and credit card information screen (infoCreditCard). The user only sees the login screen.

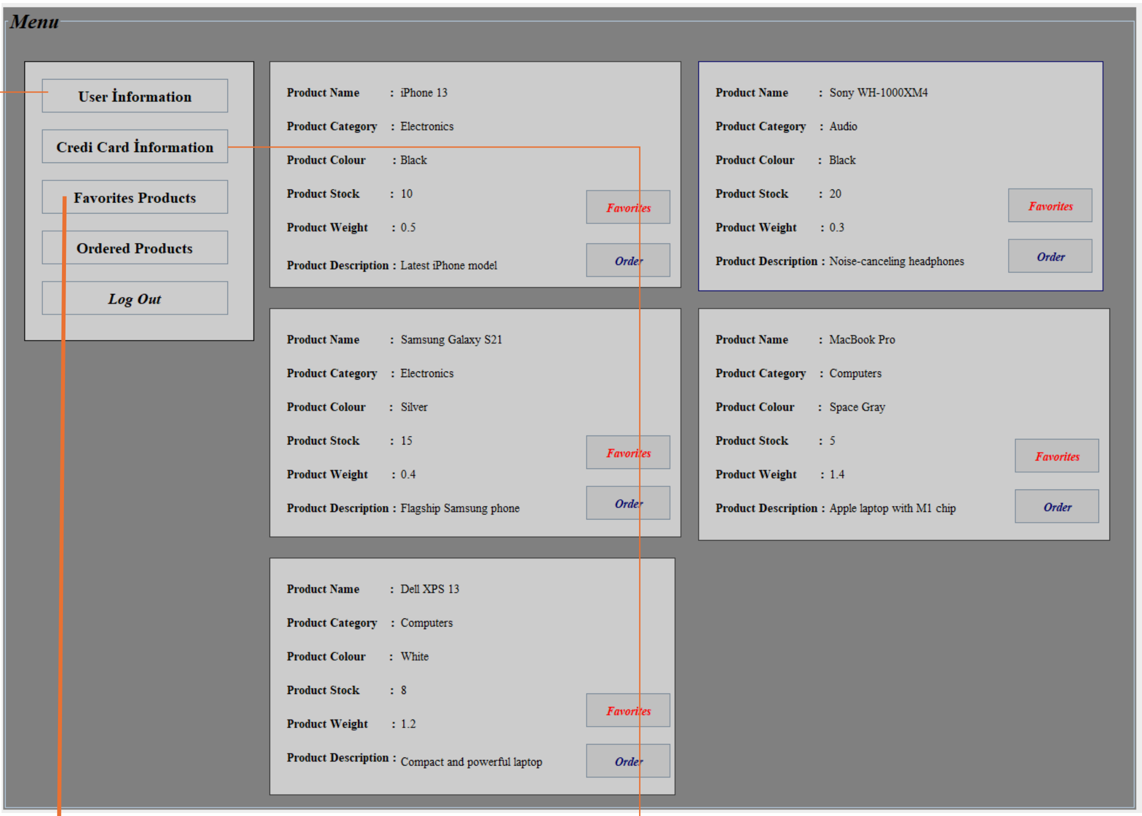
These lines clear the username (txtusername) and password (txtpassword) fields. It also clears any warning messages (warnings). Thus, the user encounters a clean screen when making a new login.

These lines reset the user's favorite product count (favoriteProductCount) and ordered product count (orderedProductCount). When the user logs in again, these numbers start from zero.

OUTPUT







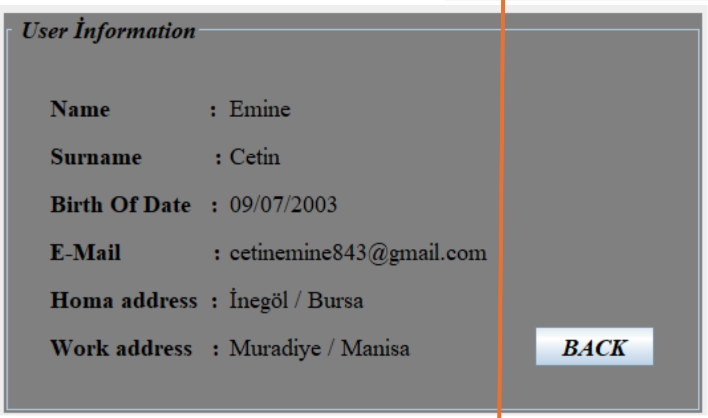
Product Name : iPhone 13
Product Category : Electronics
Product Colour : Black
Product Stock : 10
Product Weight : 0.5
Product Description : Latest iPhone model

Product Name : Sony WH-1000XM4
Product Category : Audio
Product Colour : Black
Product Stock : 20
Product Weight : 0.3
Product Description : Noise-canceling headphones

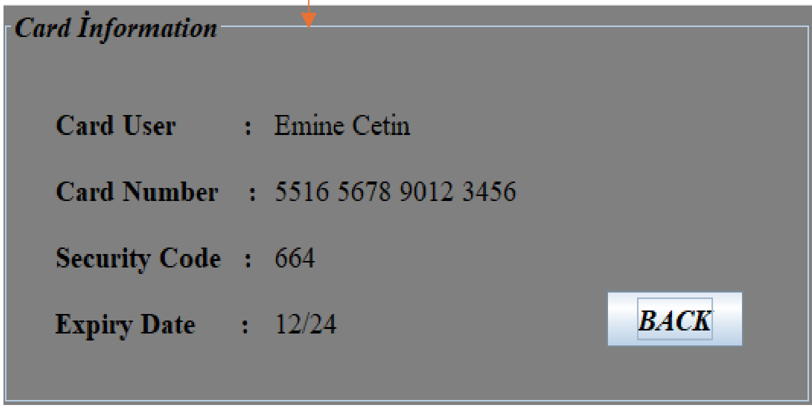
Product Name : Samsung Galaxy S21
Product Category : Electronics
Product Colour : Silver
Product Stock : 15
Product Weight : 0.4
Product Description : Flagship Samsung phone

Product Name : MacBook Pro
Product Category : Computers
Product Colour : Space Gray
Product Stock : 5
Product Weight : 1.4
Product Description : Apple laptop with M1 chip

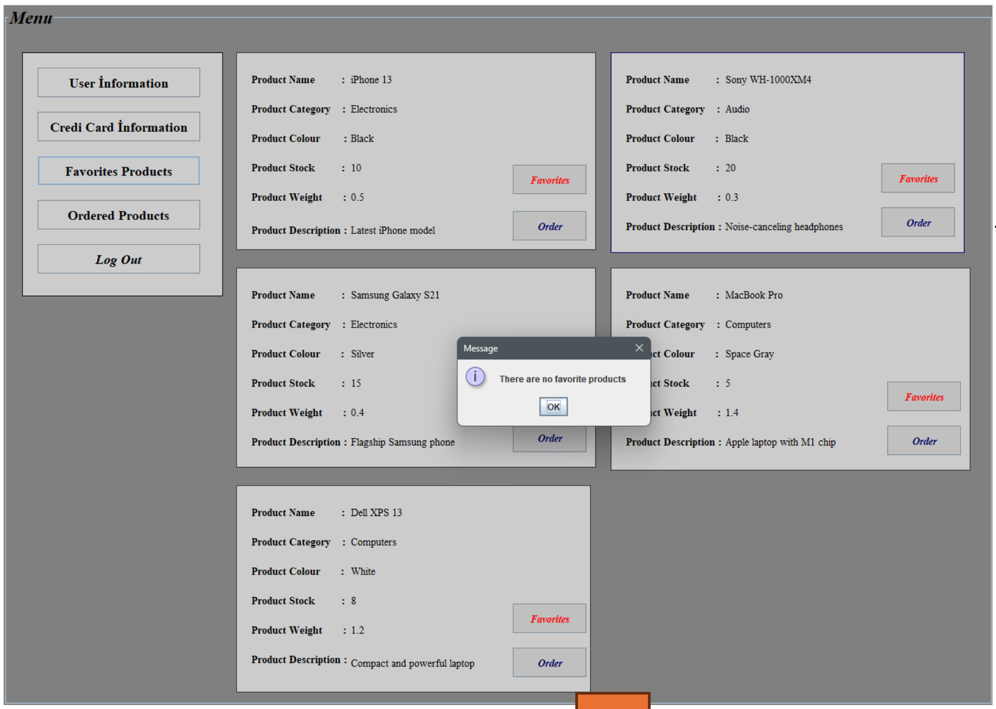
Product Name : Dell XPS 13
Product Category : Computers
Product Colour : White
Product Stock : 8
Product Weight : 1.2
Product Description : Compact and powerful laptop



Name : Emine
Surname : Cetin
Birth Of Date : 09/07/2003
E-Mail : cetinemine843@gmail.com
Homa address : İnegöl / Bursa
Work address : Muradiye / Manisa



Card User : Emine Cetin
Card Number : 5516 5678 9012 3456
Security Code : 664
Expiry Date : 12/24



Product Name : iPhone 13
Product Category : Electronics
Product Colour : Black
Product Stock : 10
Product Weight : 0.5
Product Description : Latest iPhone model

Product Name : Sony WH-1000XM4
Product Category : Audio
Product Colour : Black
Product Stock : 20
Product Weight : 0.3
Product Description : Noise-canceling headphones

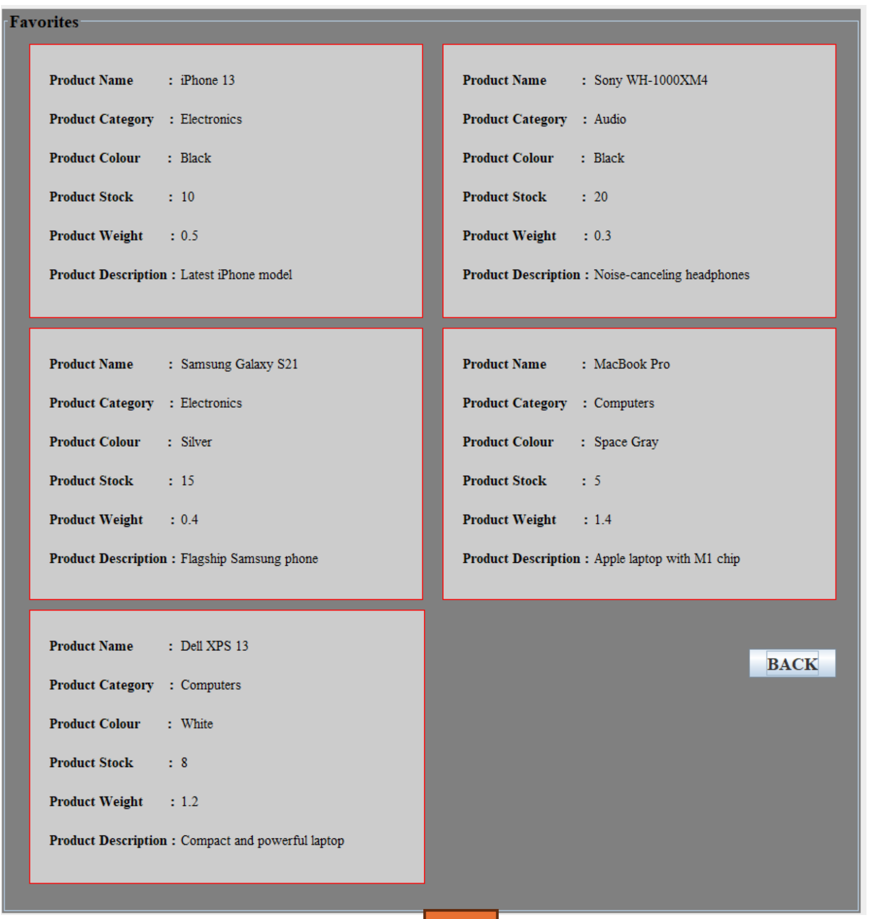
Product Name : Samsung Galaxy S21
Product Category : Electronics
Product Colour : Silver
Product Stock : 15
Product Weight : 0.4
Product Description : Flagship Samsung phone

Product Name : MacBook Pro
Product Category : Computers
Product Colour : Space Gray
Product Stock : 5
Product Weight : 1.4
Product Description : Apple laptop with M1 chip

Product Name : Dell XPS 13
Product Category : Computers
Product Colour : White
Product Stock : 8
Product Weight : 1.2
Product Description : Compact and powerful laptop

There are no favorite products

If there is no favorite product, a message appears on the screen, and it returns to the menu.


Favorites

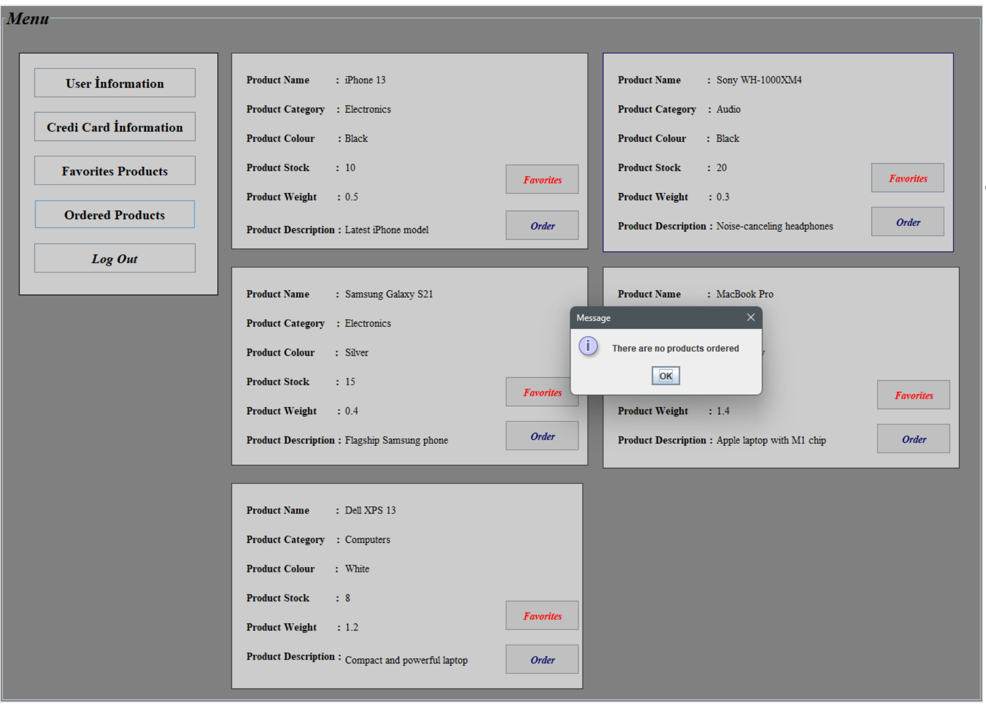
Product Name : iPhone 13
Product Category : Electronics
Product Colour : Black
Product Stock : 10
Product Weight : 0.5
Product Description : Latest iPhone model

Product Name : Sony WH-1000XM4
Product Category : Audio
Product Colour : Black
Product Stock : 20
Product Weight : 0.3
Product Description : Noise-canceling headphones

Product Name : Samsung Galaxy S21
Product Category : Electronics
Product Colour : Silver
Product Stock : 15
Product Weight : 0.4
Product Description : Flagship Samsung phone

Product Name : MacBook Pro
Product Category : Computers
Product Colour : Space Gray
Product Stock : 5
Product Weight : 1.4
Product Description : Apple laptop with M1 chip

Product Name : Dell XPS 13
Product Category : Computers
Product Colour : White
Product Stock : 8
Product Weight : 1.2
Product Description : Compact and powerful laptop


Menu

Product Name : iPhone 13
Product Category : Electronics
Product Colour : Black
Product Stock : 10
Product Weight : 0.5
Product Description : Latest iPhone model

Product Name : Sony WH-1000XM4
Product Category : Audio
Product Colour : Black
Product Stock : 20
Product Weight : 0.3
Product Description : Noise-canceling headphones

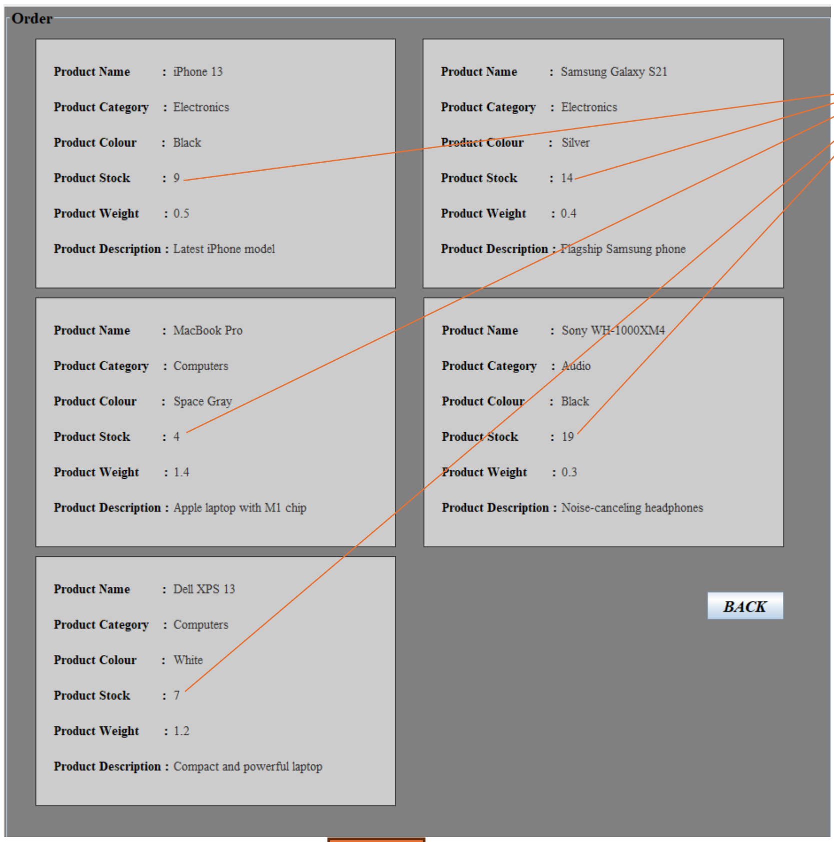
Product Name : Samsung Galaxy S21
Product Category : Electronics
Product Colour : Silver
Product Stock : 15
Product Weight : 0.4
Product Description : Flagship Samsung phone

Product Name : MacBook Pro
Product Weight : 1.4
Product Description : Apple laptop with M1 chip

Product Name : Dell XPS 13
Product Category : Computers
Product Colour : White
Product Stock : 8
Product Weight : 1.2
Product Description : Compact and powerful laptop

Message
There are no products ordered

If there is no order product, a message appears on the screen, and it returns to the menu.

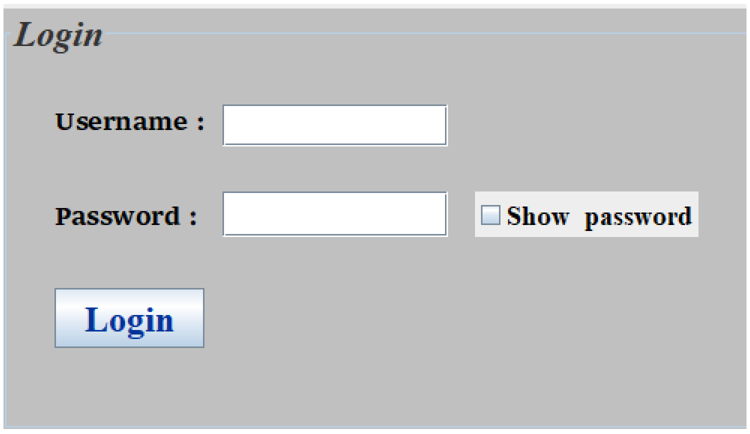


Order

Product Name : iPhone 13	Product Name : Samsung Galaxy S21
Product Category : Electronics	Product Category : Electronics
Product Colour : Black	Product Colour : Silver
Product Stock : 9	Product Stock : 14
Product Weight : 0.5	Product Weight : 0.4
Product Description : Latest iPhone model	Product Description : Flagship Samsung phone
Product Name : MacBook Pro	Product Name : Sony WH-1000XM4
Product Category : Computers	Product Category : Audio
Product Colour : Space Gray	Product Colour : Black
Product Stock : 4	Product Stock : 19
Product Weight : 1.4	Product Weight : 0.3
Product Description : Apple laptop with M1 chip	Product Description : Noise-canceling headphones
Product Name : Dell XPS 13	
Product Category : Computers	
Product Colour : White	
Product Stock : 7	
Product Weight : 1.2	
Product Description : Compact and powerful laptop	

The number of stocks of ordered products decreases.

Click the Log out button.


Login
Username :
Password :
Show password
Login