



ONLINE CASE SHOPPING SYSTEM

ARİFE GÜL YALÇIN

B1605.090054

CONTENTS

ABSTRACT	4
1. INTRODUCTION	4
2. BASIC INFORMATION	4
3. MAIN PART	5
3.1. Introduction to the Application.....	5
3.1.1. Observer Design Pattern	5
3.1.2. getDate ().....	6
3.2. Login Page.....	6
3.2.1. Register.....	7
3.2.2. Login	8
3.3. Main Menu	9
3.3.1. Home	10
3.3.2. Categories.....	10
3.3.2.1. Factory Design Pattern	10
3.3.2.2. iPhone 11.....	11
3.3.2.2.1. Flowery.....	12
3.3.2.2.2. Funny Cartoon.....	13
3.3.2.2.3. Girl Power.....	14
3.3.2.3. iPhone 11 Pro	15
3.3.2.3.1. Flowery.....	16
3.3.2.3.2. Plain Case	17
3.3.2.3.3. Girl Power.....	18
3.3.3. Shopping Card	19
3.3.4. Payment.....	20
3.3.4.1. Strategy Design Pattern.....	20
3.3.4.2. PayPal Payment Method	21
3.3.4.3. Visa Payment Method	22
3.3.4.4. Cash Payment Method	23
3.3.5. Customer Information.....	25
3.3.5.1. Builder Design Pattern.....	25
4. DATABASE.....	27
4.1. Singleton Pattern.....	27
5. CLASS DIAGRAMS	30
5.1. Observer Design Pattern	30
5.2. Singleton Design Pattern	31

5.3.	Factory Design Pattern	31
5.4.	Builder Design Pattern.....	32
5.5.	Strategy Design Pattern.....	32
6.	REFERENCES	33

SUMMARY

Online Case Shopping application is an object oriented management based application written using java where the customer can log in to the system, add the phone cases they want to the basket and then purchase the product from the payment section. A desktop application has been prepared for the Online Case Shopping Application. Gui (java swing) is used for this. The application offers 2 different models, namely categories, of 1 brand. Here, you can add the products to the cart and select the payment option. There is also an 'Information' option where the customer can enter their information. Here, when the customer, that is, the user using the application, enters the information, this information will be written to the txt file named 'info'.

ABSTRACT

The biggest purpose of the Online Case Shopping Application is to make it easier for the customer to purchase the product they want and to choose the payment option they want. In addition, since most of the information is kept in the database, it increases the speed of the customer's transactions. The fact that it is an Object Oriented Management program also makes it easier to use.

1. INTRODUCTION

The main things to know about Online Case Shopping Application are;

As soon as the user logs into the system, the date the customer logs into the application and a warning message ('WELCOME') appears on the screen. If the user is not a member of the application, customer must register, if registered, customer must log in to the application. After logging into the system, the customer can access the products, add the products customer likes to the basket, then select the application according to the desired payment type and complete the payment. In addition, customer information can be easily accessed later by entering customer information and saving customer information in a file by the system.

2. BASIC INFORMATION

Online Case Shopping Application is written using java software language. NetBeans has used IDE 8.2 for this. Customer information is written in the "Info.txt" text file. All the remaining operations of the program are done using the database. The database used for these operations;

With XAMPP Control Panel v3.2.4, Apache and MySQL are activated and phpMyAdmin is logged. Then tables are created in the database and a connection is made with the written java code.

3. MAIN PART

3.1. Introduction to the Application

3.1.1. Observer Design Pattern

Observer is a behavioral design pattern. In this design pattern there is an object called subject. Observer pattern defines a one-to-many relationship between a set of objects.

In this application, when the user enters the system, observer pattern is used. The reason for this is to observe the date the customer entered the application on the screen as in Figure 3.1.1.

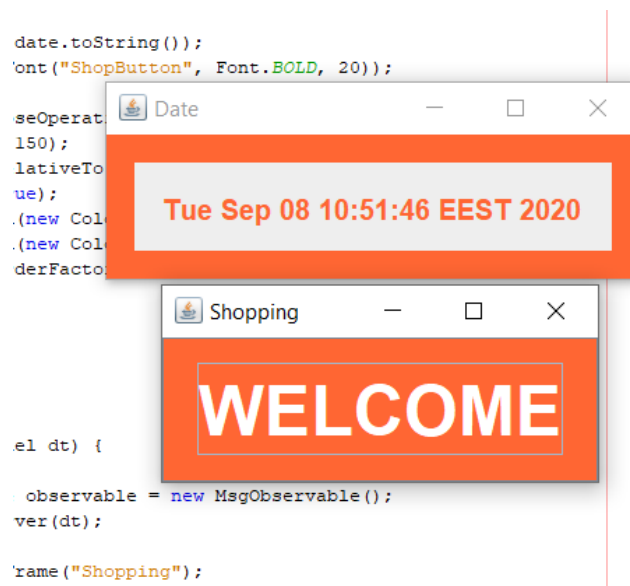


Figure 3.1.1. Observer Design Pattern (Introduction to the Application)

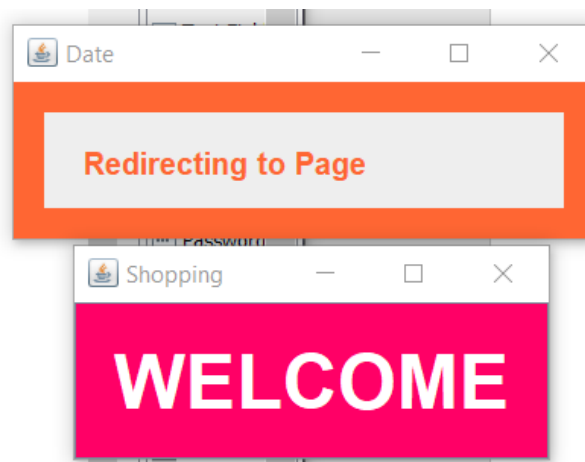


Figure 3.1.1.1. Observable (Introduction to the Application)

3.1.2. getDate ()

Firstly, 'java.util.Date' is imported or otherwise, all can be imported with ' java.util. * '.

Later, an object named date was defined from the Date class. In the created label, the object was called by calling 'date.toString ()'.

3.2. Login Page

When the customer enters the Login Page page, if he has previously created a membership, that is, if he is a registered user in the database, customer can log into the application via the 'Login' option. But if the customer is the first login, he / she must first be a member from the 'Register' option, as in Figure 3.2.

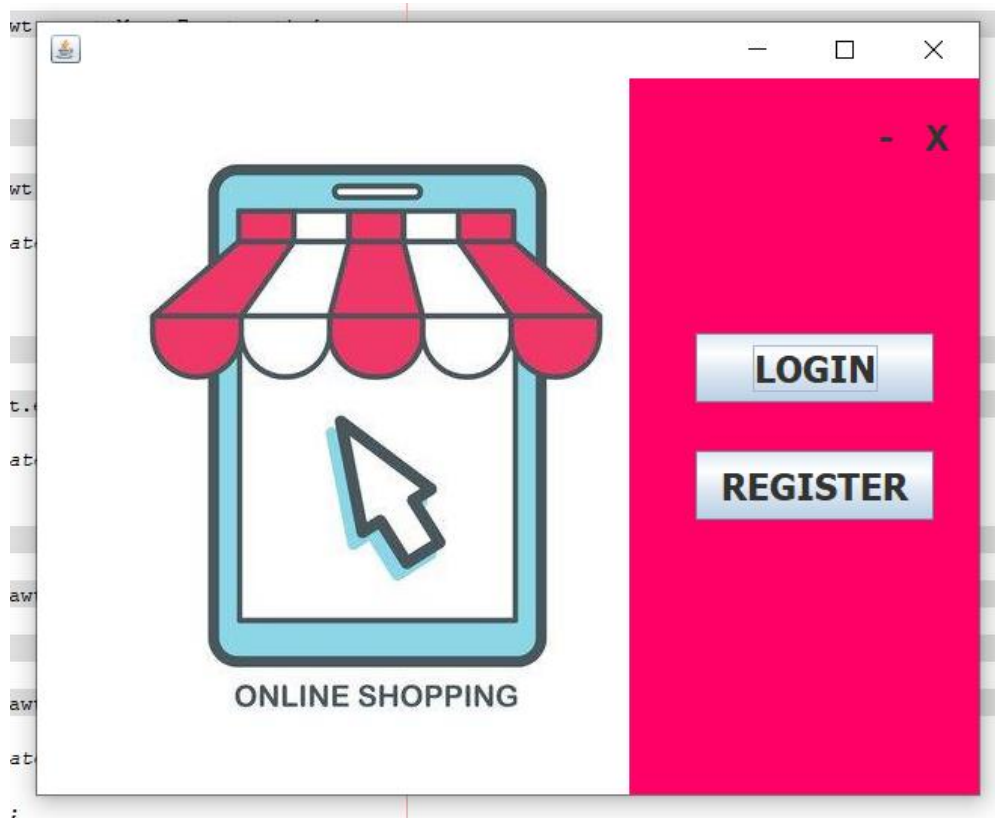


Figure 3.2. Login Application

3.2.1. Register

The customer must enter some information requested from the user for membership in the relevant fields. In addition, depending on the customer's request, the application should include a photo adding button.

A screenshot of a web application's 'REGISTER' page. The page has a light blue header with a purple 'REGISTER' button. Below the header is a light pink form area. The form contains several input fields: 'Full Name:', 'Username:', 'Password:', 'Confirm Password:', 'Phone:', and 'Gender:' with radio buttons for 'Female' and 'Male'. There is also an 'Image:' section with a 'Select Image' button and a text field for 'image path'. At the bottom of the form is a large red 'Register' button and a '>> Login' link.

Figure 3.2.1. Register Page

If the customer leaves the relevant fields more than once, the user receives a warning message, as in Figure 3.2.1.1.

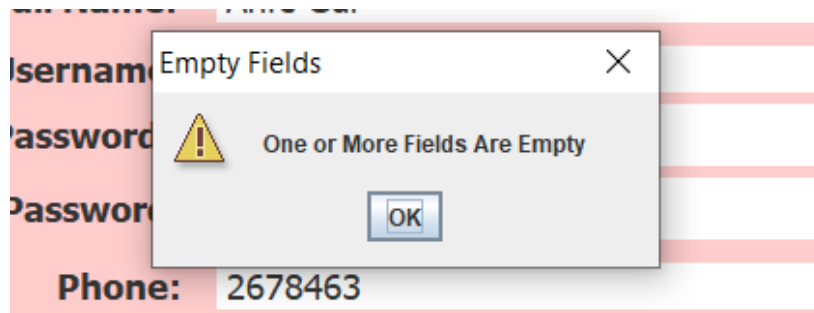


Figure 3.2.1.1. Empty Error Message

If the customer does not enter the same thing as the first password in the confirm password section, the user will receive an error message.

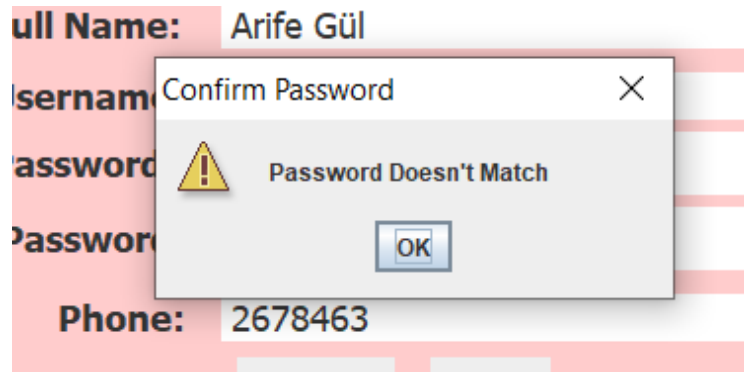


Figure 3.2.1.2. Password Error Message

If the customer wants to use a username that was previously in the database, customer will get an error message. The user must choose a unique username that is not in the database.

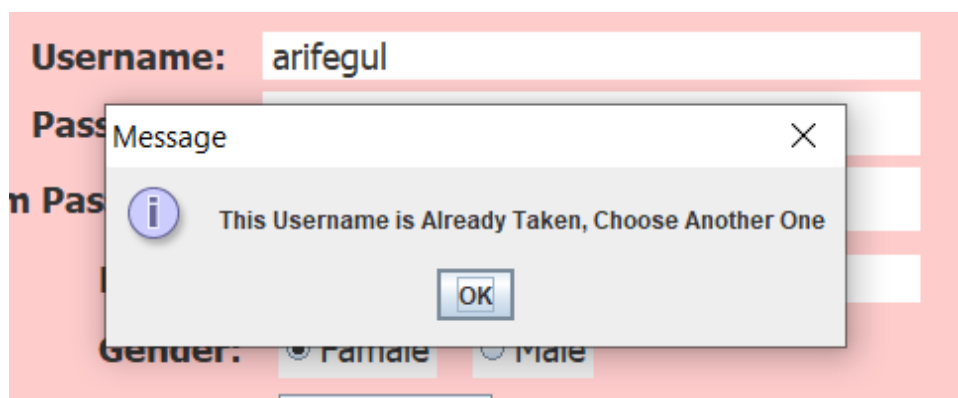


Figure 3.2.1.3. Username Error Message

3.2.2. Login

If the customer has registered to the system before, customer will be able to enter the application using their username and password. This information is included in the 'users' table in the database.

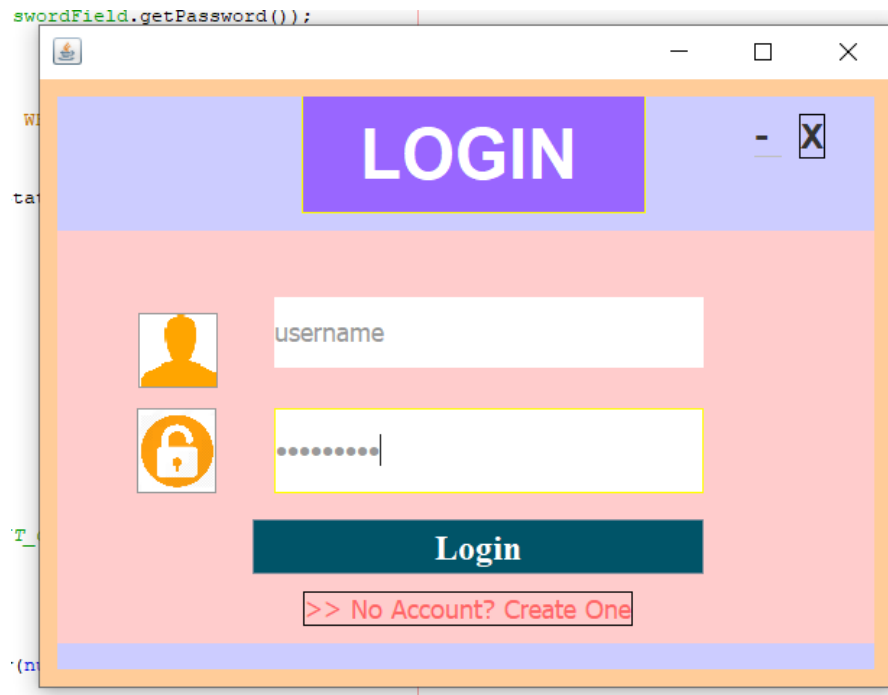


Figure 3.2.2. Login

3.3. Main Menu

There are options for the customer to easily access the application. Whichever of these options the customer wants to reach, it is possible to access that page.

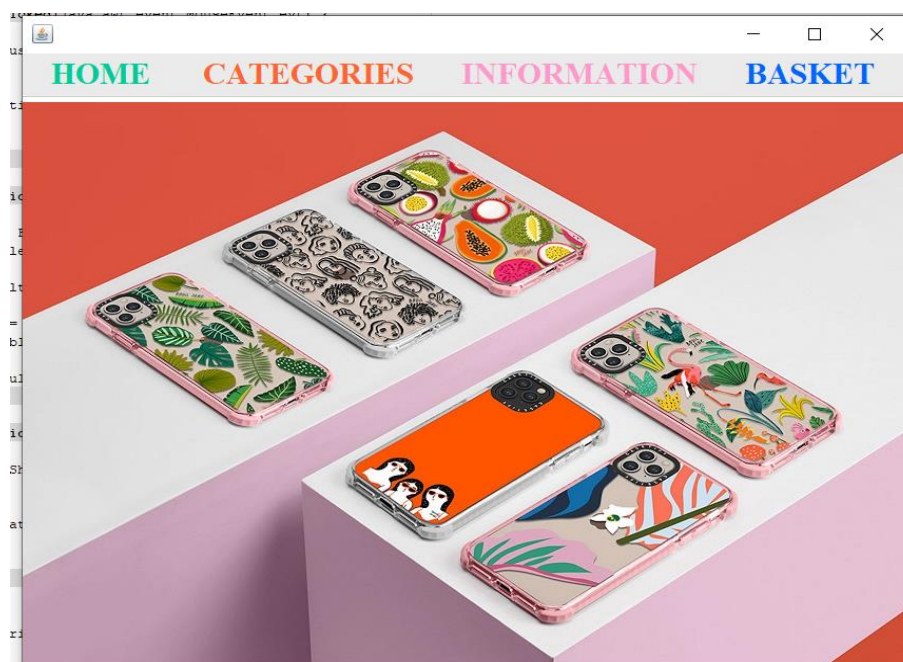


Figure 3.3. Main Menu

3.3.1. Home

It is an option for the customer to refresh their 'Main Menu' page.

3.3.2. Categories

There are 2 categories for customer online shopping.

3.3.2.1. Factory Design Pattern

'Factory Design Pattern' is used in the categories section. Here are the things done;

- ✓ First, a 'Brand' class was created. Here, a void displayName () and JFrame createWindow () are defined.
- ✓ Things defined in the Brand class will be @override in the iphone.java and iphonepro.java classes.
- ✓ Frames created in iphone.java and iphonepro.java classes represent two models of a brand.
- ✓ So here the iPhone 11 and iPhone 11 Pro categories created from the Apple factory will be displayed.
- ✓ Frames previously defined as createWindow () in Factory.java class are received with getFrame ().
- ✓ Finally, with 2 frames MainClass.java created for Factory Design Pattern, frames are printed by creating a new object in main. This is shown on the screen with setVisible (true).



Figure 3.3.2.1. Categories

3.3.2.2. iPhone 11

There are phone case models that the customer can choose from the iPhone 11 category.



Figure 3.3.2.2. iPhone 11 Phone Categories

3.3.2.2.1. Flowery

In this section, the customer will be able to see the iPhone 11 Flowery phone case models.



Figure 3.3.2.2.1. iPhone 11 Flowery

3.3.2.2.2. Funny Cartoon

In this section, the customer will be able to see the iPhone 11 Funny Cartoon phone case models.



FUNNY CARTOON



F1

CASEID:

PRICE:





F2

CASEID:

PRICE:





F3

CASEID:

PRICE:





F4

CASEID:

PRICE:





F5

CASEID:

PRICE:





F6

CASEID:

PRICE:



<<- BACK

SHOPPING CART ->>

Figure 3.3.2.2.2. iPhone 11 Funny Cartoon

3.3.2.2.3. Girl Power

In this section, the customer will be able to see the iPhone 11 Girl Power phone case models.



Figure 3.3.2.2.3. iPhone 11 Girl Power

3.3.2.3. iPhone 11 Pro

There are phone case models that the customer can choose from the iPhone 11 category.



Figure 3.3.2.3. iPhone 11 Pro Categories

3.3.2.3.1. Flowery

In this section, the customer will be able to see the iPhone 11 Pro Flowery phone case models.



Figure 3.3.2.3.1. iPhone 11 Pro Flowery

3.3.2.3.2. Plain Case

In this section, the customer will be able to see the iPhone 11 Pro Plain Case phone case models.



Figure 3.3.2.3.2. iPhone 11 Pro Plain Case

3.3.2.3.3. Girl Power

In this section, the customer will be able to see the iPhone 11 Pro Girl Power phone case models.



Figure 3.3.2.3.3. iPhone 11 Pro Girl Power

3.3.3. Shopping Card

After the customer adds the products to the basket, the orders of the customer are shown in the basket table in the database.

After selecting the products, the customer can click on the shopping cart and switch to the 'Shopping Card' page.

When the customer comes to this page, they can access their orders with the 'shopping card' button.

Later, the customer should select the orders one by one and click the 'Add Basket' button.

Here, the customer will be able to see the total prices of all products selected by the customer.

The customer will be able to cancel adding the order with the 'Cancel' button and continue adding other orders from where they left off.

Then, in order to finish the order, the customer must enter the username he used when entering the application in the last part specified as username.

The customer can complete the order by clicking the 'Finish Order' button and proceed to the payment section.

id	caselid	price
15	F2	40
16	F3	40

SHOPPING CARD

F3 40

80.0

Add Basket Cancel

<< MENU User Name arifegul Finish Order PAYMENT ->>

Figure 3.3.3. Shopping Card

3.3.4. Payment

3.3.4.1. Strategy Design Pattern

The strategy design template is used to select and apply a method according to the situation. Each method (algorithm) is implemented for a class. So if we add up the issue, the behavior or algorithm of a class at runtime can be changed according to a strategy.

In order to apply the strategy design template, we need to define an interface named Strategy. This interface contains methods or methods that will be implemented by subclasses.

The strategy applied for this application is related to the customer's payment method. First, a PaymentMethod.java script is created. This is defined as the class interface. PayPal.java, Visa.java and Cash.java classes have been created for payment methods. In addition, information to be requested from the customer is defined in each class created.

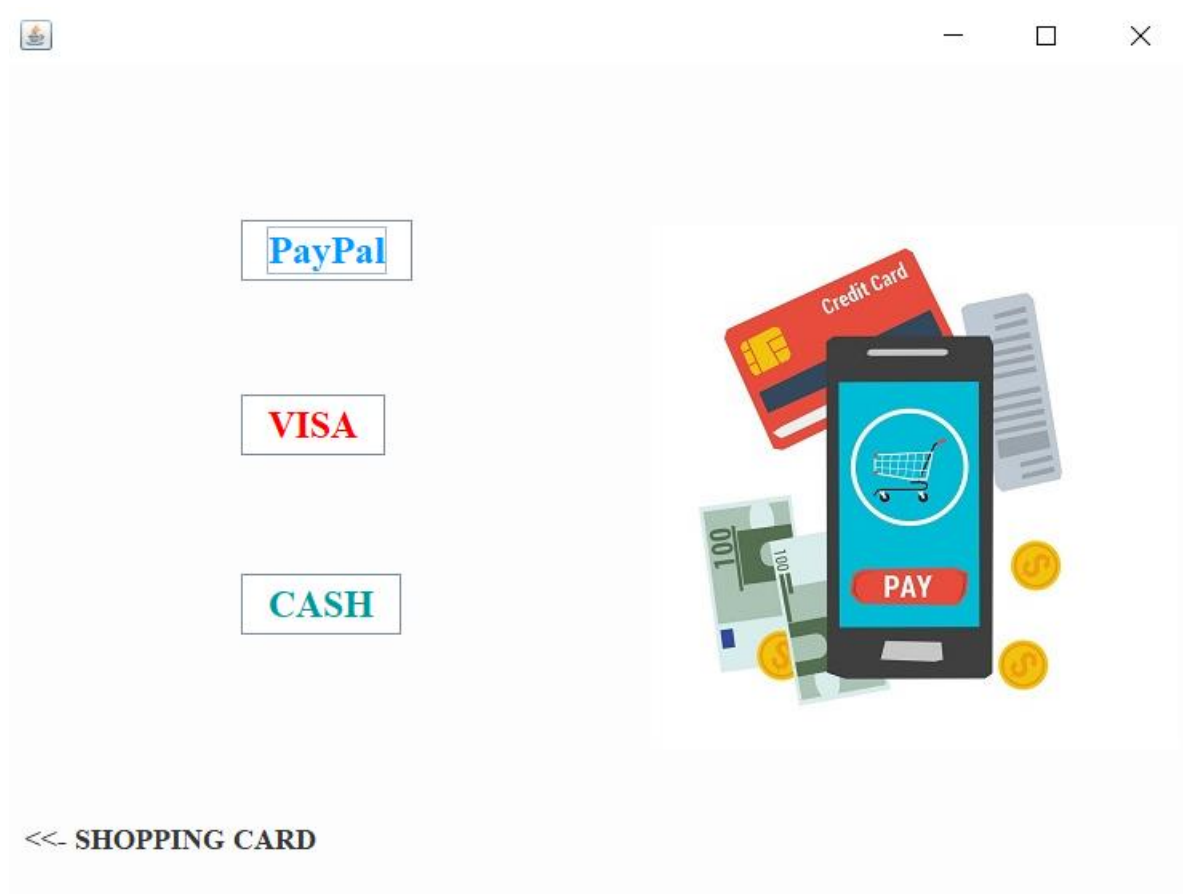


Figure 3.3.4. Paymnet Method (Strategy Design Pattern)

3.3.4.2. PayPal Payment Method

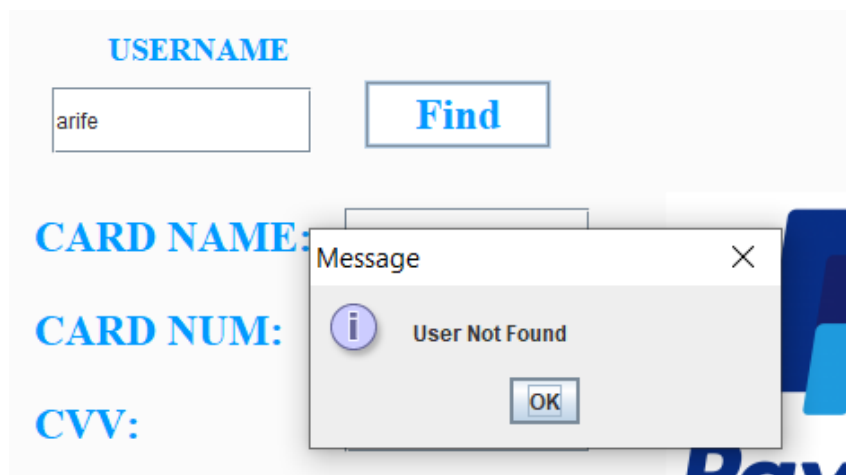
In order to find out which order the customer has in the database, username is requested and the 'Find' button is clicked. The field specified as Price will be able to access how much the customer will pay. Then the customer must fill in the remaining blank fields. The fields to be filled by the customer are shown in figure 3.3.4.2. as in.



The screenshot shows a web form titled "PayPal Payment Method". At the top, there is a "USERNAME" label above a text input field containing "arifegul". To the right of this field is a blue "Find" button. Below the username field, there are five more input fields, each with a label to its left: "CARD NAME:", "CARD NUM:", "CVV:", "EXPIRES:", and "PRICE:". The "PRICE:" field contains the value "80". To the right of these fields is a large PayPal logo. At the bottom left of the form is a "<< BACK" link, and at the bottom right is a blue "PAY" button.

Figure 3.3.4.2. Paypal Payment Method

If the customer cannot view an order when entering their username, they will receive a warning message.

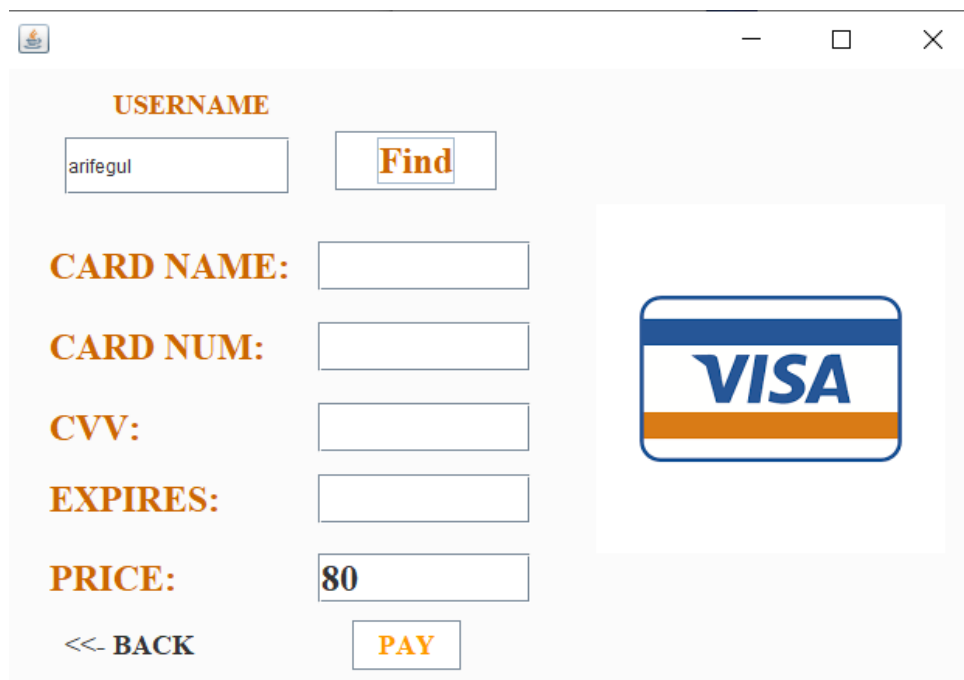


This screenshot shows the same PayPal Payment Method form as Figure 3.3.4.2, but with a warning message overlay. The "USERNAME" field now contains "arife". A modal dialog box titled "Message" is centered over the form. It features an information icon (i) and the text "User Not Found". There is an "OK" button at the bottom right of the dialog box. The background form is partially obscured by the dialog box.

Figure 3.3.4.2.1. Paypal Payment Method Warning Message

3.3.4.3. Visa Payment Method

In order to find out which order the customer has in the database, username is requested and the 'Find' button is clicked. The field specified as Price will be able to access how much the customer will pay. Then the customer must fill in the remaining blank fields. The fields to be filled by the customer are shown in figure 3.3.4.3. as in.

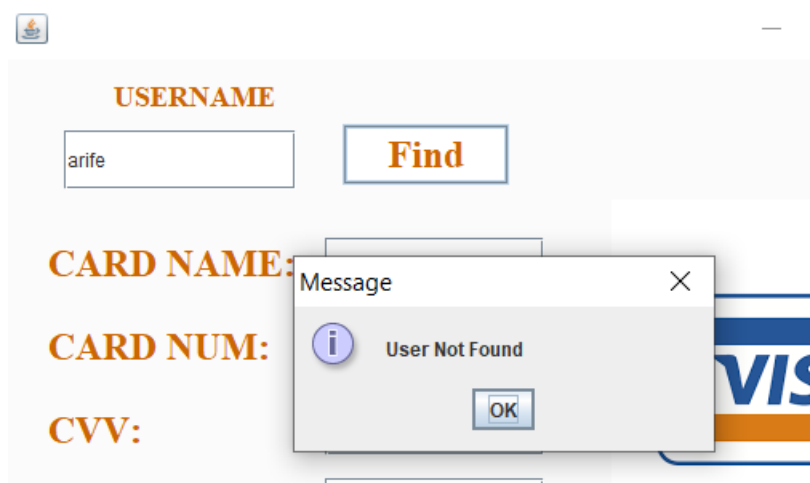


The screenshot shows a web application window titled "Visa Payment Method". It contains a form with the following elements:

- USERNAME:** A text input field containing "arifegul" and a "Find" button.
- CARD NAME:** A text input field.
- CARD NUM:** A text input field.
- CVV:** A text input field.
- EXPIRES:** A text input field.
- PRICE:** A text input field containing "80".
- Navigation:** A "<< BACK" button and a "PAY" button.
- Image:** A Visa logo is displayed on the right side of the form.

Figure 3.3.4.3. Visa Payment Method

If the customer cannot view an order when entering their username, they will receive a warning message.

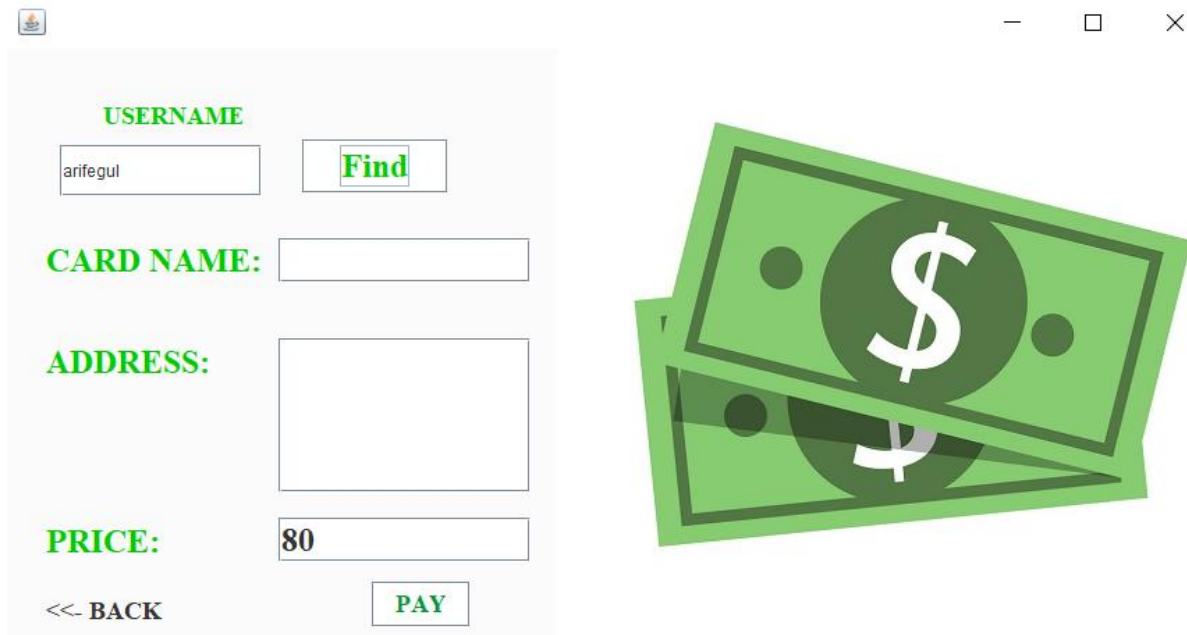


The screenshot shows the same "Visa Payment Method" form as in Figure 3.3.4.3, but with a warning message displayed over it. The message box is titled "Message" and contains the text "User Not Found" with an "OK" button. The form fields are partially obscured by the message box. The "USERNAME" field contains "arife" and the "Find" button is visible. The "CARD NAME:", "CARD NUM:", and "CVV:" labels are also visible.

Figure 3.3.4.3.1. Visa Payment Method Warning Message

3.3.4.4. Cash Payment Method

In order to find out which order the customer has in the database, username is requested and the 'Find' button is clicked. The field specified as Price will be able to access how much the customer will pay. Then the customer must fill in the remaining blank fields. The fields to be filled by the customer are shown in figure 3.3.4.4. as in.



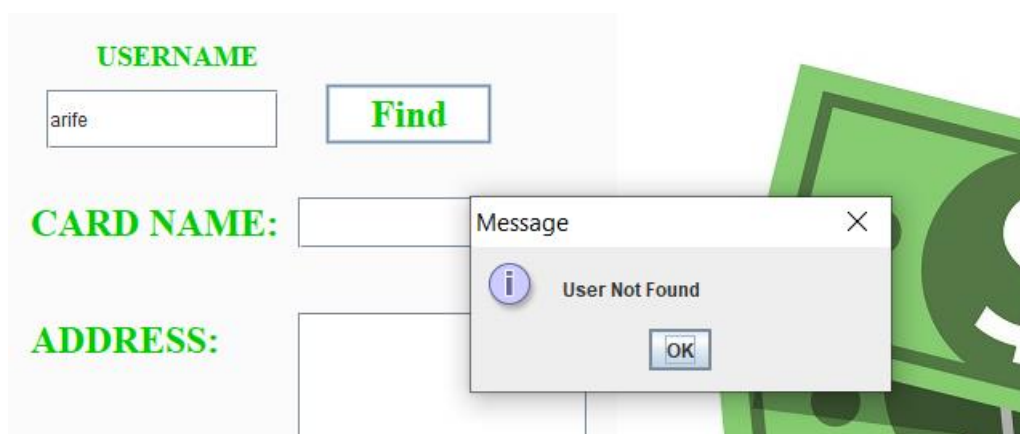
The screenshot shows a web form titled "Cash Payment Method". It contains the following fields and buttons:

- USERNAME:** A text input field containing "arifegul" and a green "Find" button.
- CARD NAME:** A text input field.
- ADDRESS:** A text input field.
- PRICE:** A text input field containing "80".
- Navigation:** A green "<< BACK" button and a green "PAY" button.

To the right of the form is a graphic of two green banknotes with a large white dollar sign.

Figure 3.3.4.4. Cash Payment Method

If the customer cannot view an order when entering their username, they will receive a warning message.



This screenshot shows the same form as Figure 3.3.4.4, but with a warning message displayed over it. The "USERNAME" field now contains "arife". The warning message box is titled "Message" and contains the text "User Not Found" with an information icon and an "OK" button.

Figure 3.3.4.4.1. Cash Payment Method Warning Message

After the customer has used one of the payment methods and entered the requested information in the relevant fields, clicking the 'Pay' button will delete the customer's order in the database as the payment process is completed.

The way it appears in the database is as follows.

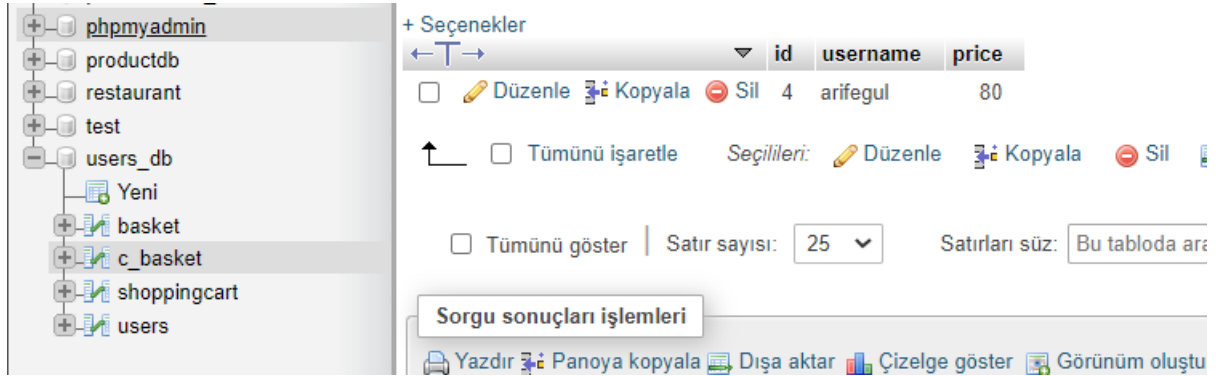


Figure 3.3.4.1. Before Payment

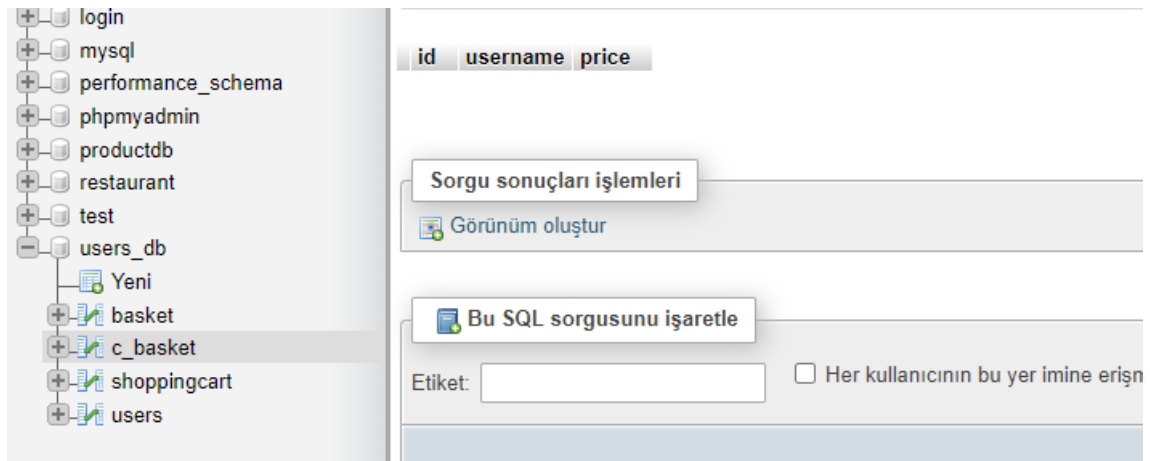


Figure 3.3.4.1.1 After Payment

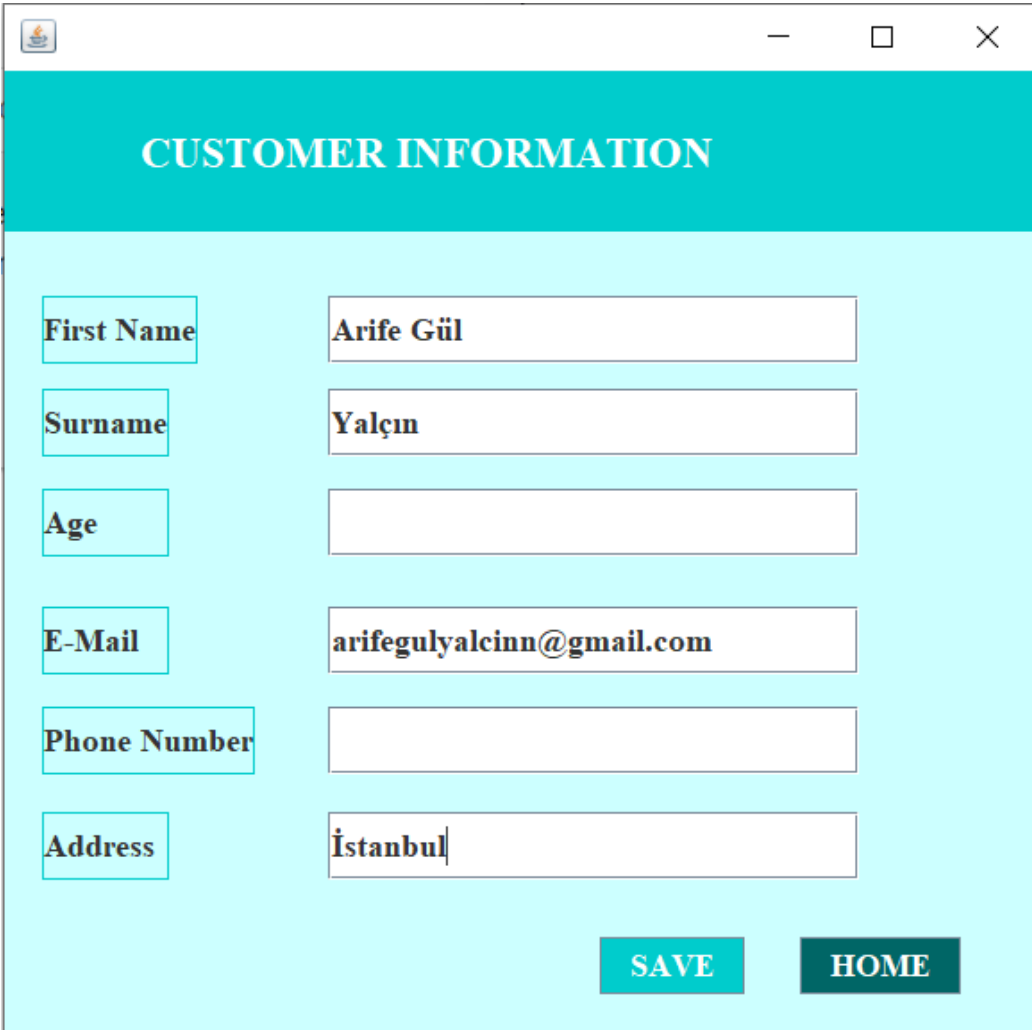
3.3.5. Customer Information

3.3.5.1. Builder Design Pattern

The essence of object-oriented programming is based on classes. Objects are created from classes. Constructors are used to do this. If the number of fields in the class is too many, therefore more than one constructor may be needed. Consequently, the need to add a new constructor may be felt when each field is added. Because while creating the object, it may not be known which field will be initially assigned or not. Builder Design Pattern is used for this.

The purpose of the Builder Design Pattern used in this application is to make the customer not enter some information optionally when he / she will enter the customer information.

The data on this page, where the customer enters the information, is written in the "Info.txt" text file.



CUSTOMER INFORMATION	
First Name	Arife Gül
Surname	Yalçın
Age	
E-Mail	arifegulyalcinn@gmail.com
Phone Number	
Address	İstanbul
<div>SAVEHOME</div>	

Figure 3.3.5. Customer Information (Builder Design Pattern)

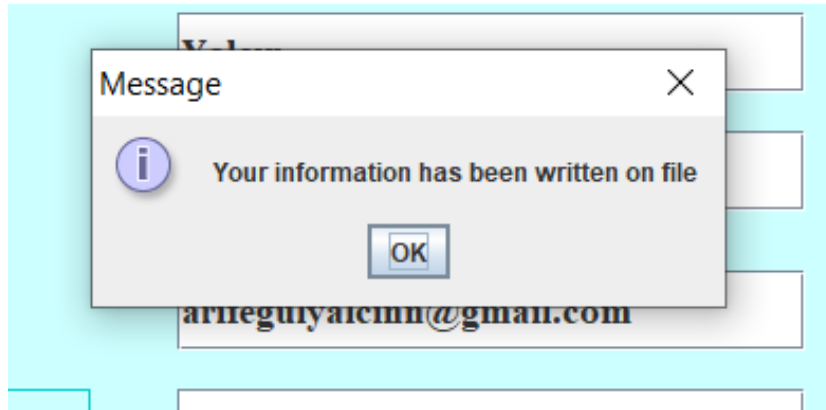


Figure 3.3.5.1. Customer Information

info - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

First Name : Arife Gül
Surname : Yalçın
Age :
E-Mail : arifegulyalcinn@gmail.com
Phone Number :
Address : İstanbul

Figure 3.3.5.2. Customer Information (info.txt)

4. DATABASE

4.1. Singleton Pattern

Singleton design pattern is used in situations where a single instance of the object is desired. It can be used for operations such as database connections, port operations.

The list of database drivers is managed with the DriverManager class.

Provides communication with driver database.

Connection provides database communication with all methods.

First, the constructor function is defined as private. And database connection processes are started.

The drivers were registered first here. Then, the database is physically connected using the DriverManager.getConnection () method whose return type is connection. In order to get an object from this class, the getInstance () method is written.

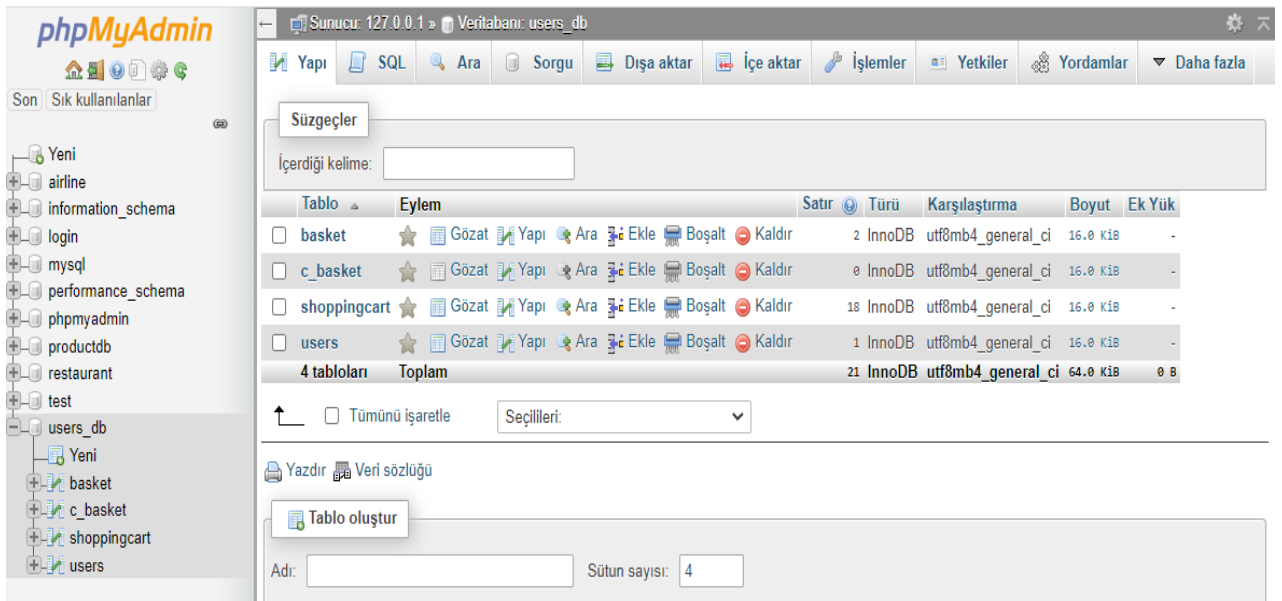


Figure 4. Database Tables

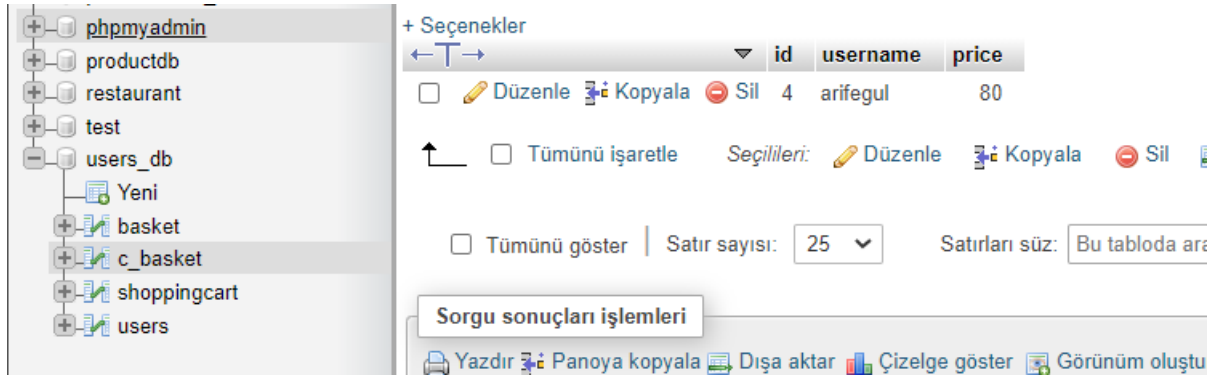


Figure 4.3. Database Payment Table

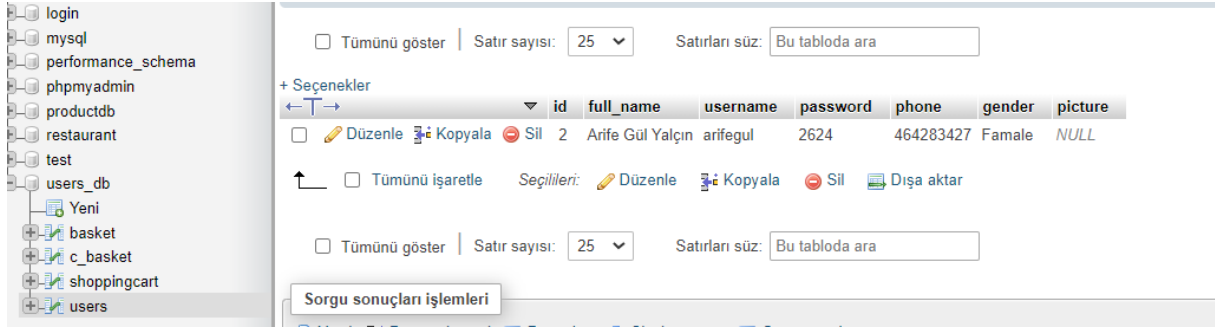
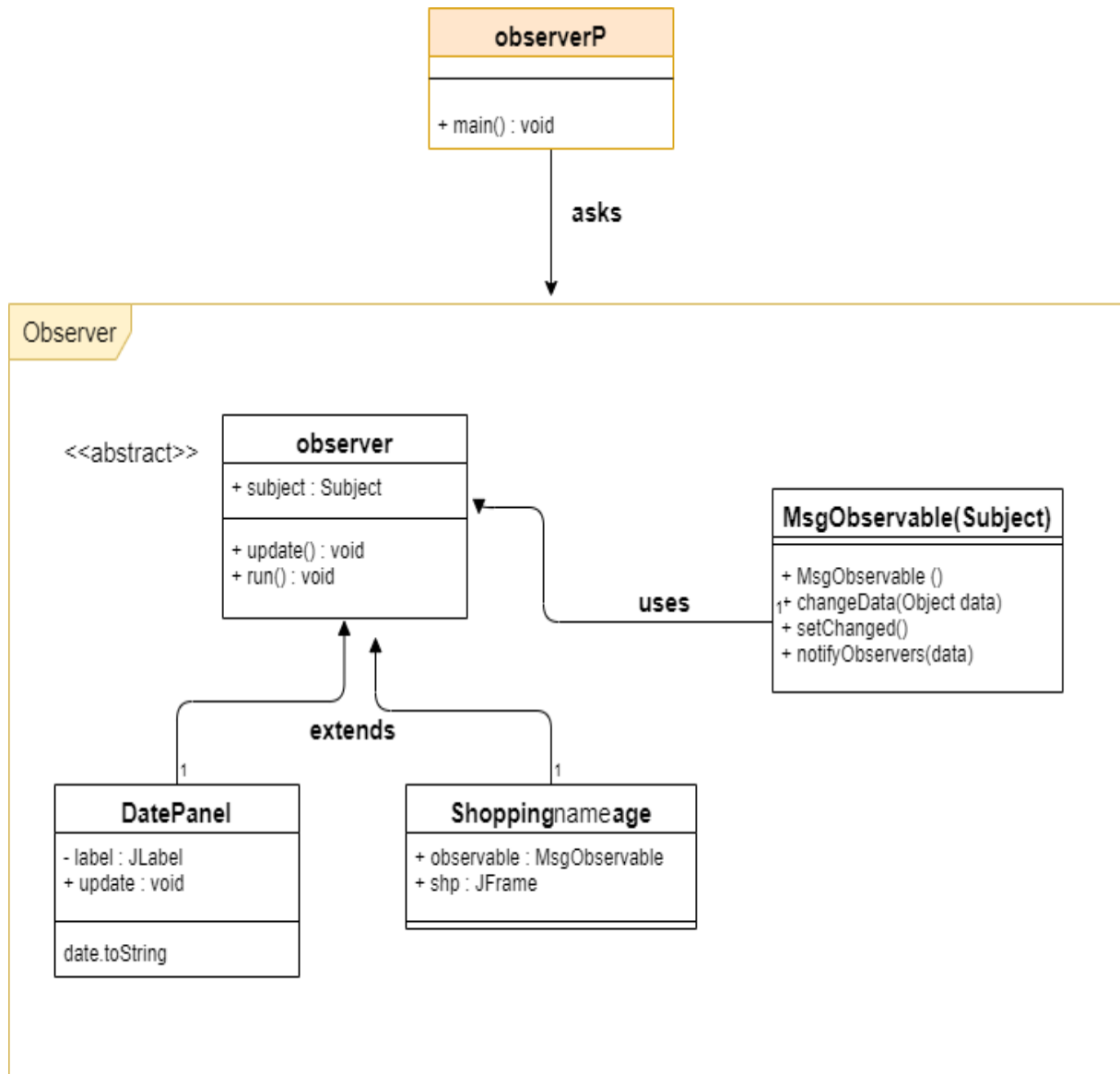


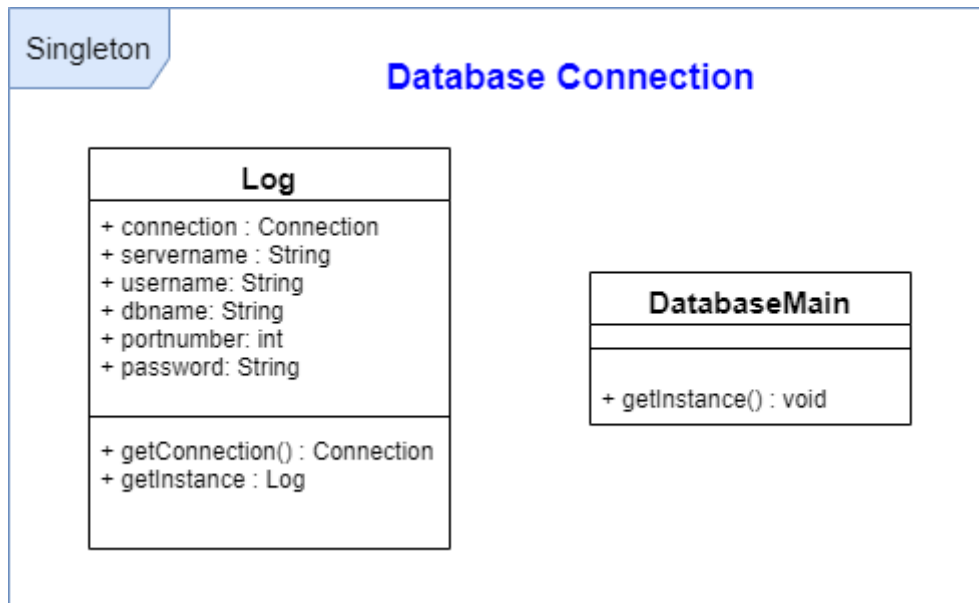
Figure 4.4. Database User Table

5. CLASS DIAGRAMS

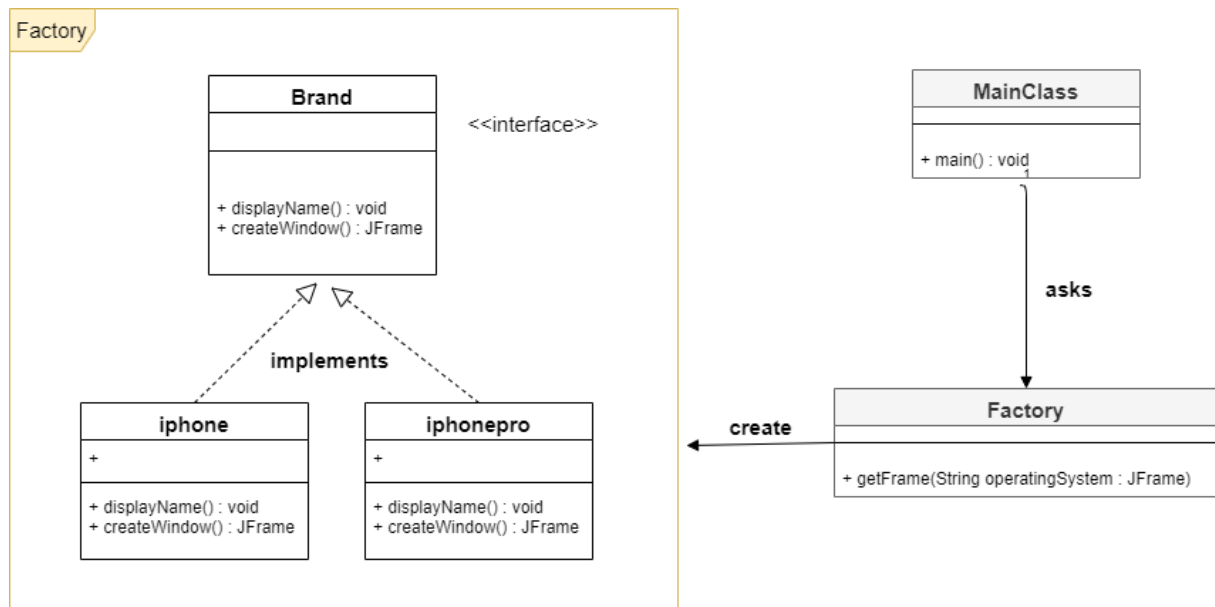
5.1. Observer Design Pattern



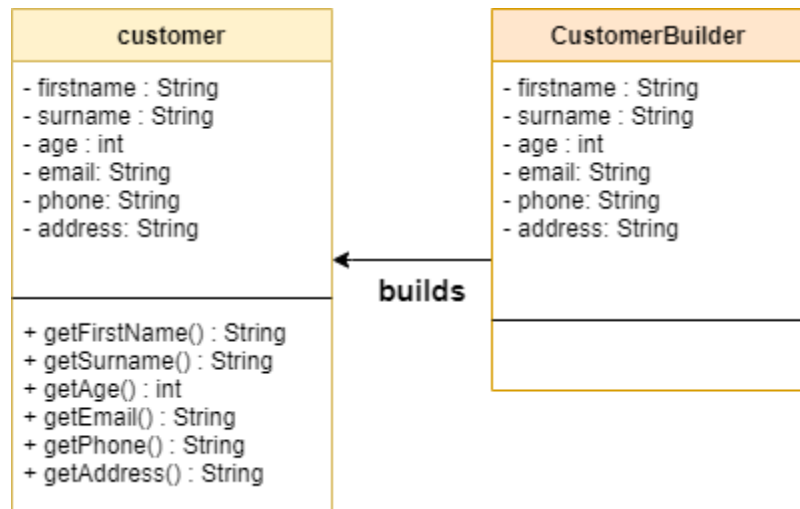
5.2. Singleton Design Pattern



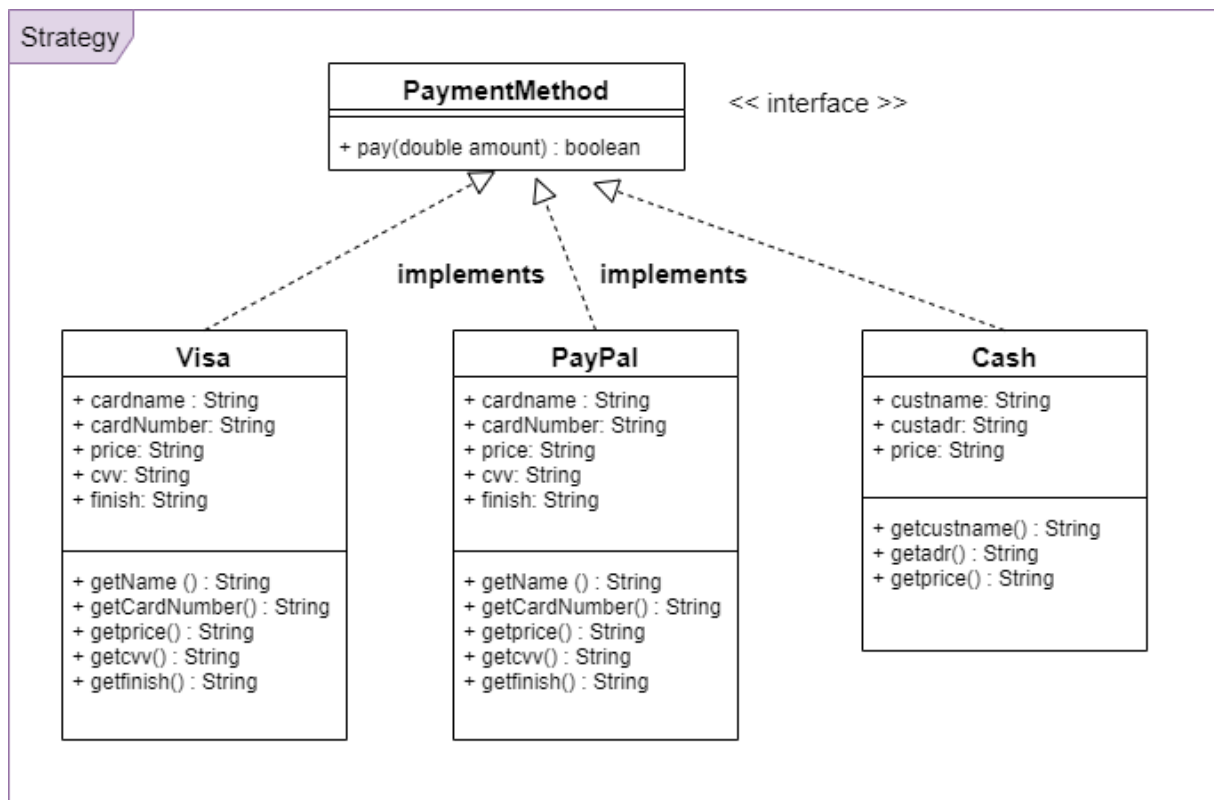
5.3. Factory Design Pattern



5.4. Builder Design Pattern



5.5. Strategy Design Pattern



6. REFERENCES

Guru99. (2015, 09 20). Guru99 Web Site:

<https://www.guru99.com/java-swing-gui.html>

Java Point. (2013, 11 28). Java Point Web Site:

<https://www.javatpoint.com/design-patterns-in-java>

Java T Point. (2017, 03 02). Java T Point Web Site:

<https://www.javatpoint.com/steps-to-connect-to-the-database-in-java>

Java Yaz. (2018, 12 10). Java Yaz Web Site:

http://javayaz.com/?page_id=107

Medium. (2020, 4 22). Medium Web Sitesi:

<https://medium.com/@kadermiyanyedi/singleton-tasar%C4%B1m-deseni-i%CC%87le-java-database-connection-9b3d460aa941>

Oracle. (2018, 10 28). Oracle Web Site:

<https://docs.oracle.com/javase/tutorial/jdbc/basics/jdbcswing.html>

Tutorial Point. (2019, 11 14). Tutorial Point Web Sitesi:

https://www.tutorialspoint.com/design_pattern/index.htm