

# AREL GİYİM MANAGEMENT SYSTEM DESIGN DOCUMENT V.1.1

## DOCUMENT REVISION HISTORY

Date	Version	Description	Author
20/02/2024	0.0	Initial draft	Prof.Dr.Taner Çevik
26/05/2024	0.1	Report is done	Hüseyin Kaya

## TABLE OF CONTENTS

- 1. Introduction
  - 1.1 Purpose
  - 1.2 System Overview
  - 1.3 Design Goals
  - 1.4 References
  - 1.5 Definitions, Acronyms, and Abbreviations
- 2. Design Overview
  - 2.1 Introduction
  - 2.2 Environmental Overview
  - 2.3 System Architecture
  - 2.4 Constraints and Assumptions
- 3. Interfaces and Data Storage
  - 3.1 System Interfaces
  - 3.2 Data Storage
- 4. Structural Design
  - 4.1 Design Description and Rationale
  - 4.2 Class Diagram

#### 4.3 Class Descriptions

#### 5. Dynamic Model

##### 5.1 Scenarios

#### 6. Appendices

## 1. INTRODUCTION

### 1.1 PURPOSE

This system was created for sales in Arel Giyim store.

### 1.2 SYSTEM OVERVIEW

This project includes Arel Giyim Management System software design details. The project aims to log in on behalf of the store manager, manage and view store inventory, user information and sales, register and log in for the customer, add/remove any item from the store inventory to cart, view purchases and possessions.

### 1.3 DESIGN OBJECTIVES

Arel clothing store system builds a bridge between the manager and the customer. The manager can view every product sold, and the customer can view every product added by the manager.

#### Manager

The manager can add as many products as he wants to the store, as well as as many color, size and stock combinations as he wants for any product. Can delete products in the store. He can update the price of the product(s) he wants. Can see the remaining stocks in the store. In addition, the manager can see all the products purchased by the customers so far by pressing the "Satışlar" button along with the personal information of the purchaser.

#### Customer

When the customer presses the "Add to Cart" button, all the products the manager added to the store are listed and after clicking on the product he wants, he is asked how many items he wants to add to his cart. The customer enters a numerical value and the product is added to the cart. If the customer wishes, he can query the products he added to his cart. If the customer wants to buy the product(s), he/she can press the "Buy" button and buy the product he/she wants. If the customer wishes, he can also view the products he purchased through the interface.

### 1.4 REFERENCES

Object-Oriented Programming (OOP) principles

Java Programming Language and Swing Library

[www3.ntu.edu.sg](http://www3.ntu.edu.sg)

<https://www.w3schools.com/>

ChatGPT

## 1.5 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

OOP: Object-Oriented Programming

GUI: Graphical User Interface

MVC: Model-View-Controller

## 2. DESIGN OVERVIEW

### 2.1 INTRODUCTION

This program has been entirely written according to the principles of Object-Oriented Programming (OOP). For designing and creating graphical user interfaces (GUI's) Java Swing and Eclipse is used.

### 2.2 ENVIRONMENTAL OVERVIEW

The program designed to be desktop application and the users of the program will be provided with the executable jar file of the program to run it.

### 2.3 SYSTEM ARCHITECTURE

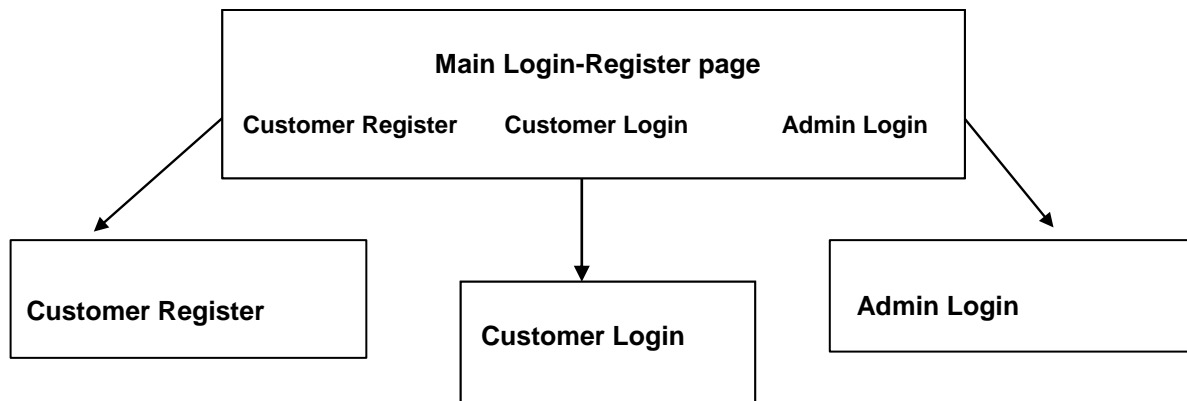
#### 2.3.1 Top-Level System Structure of Arel Giyim Management System

##### Main Login-Register Page

-Admin Login

-Customer Login

-Customer Register



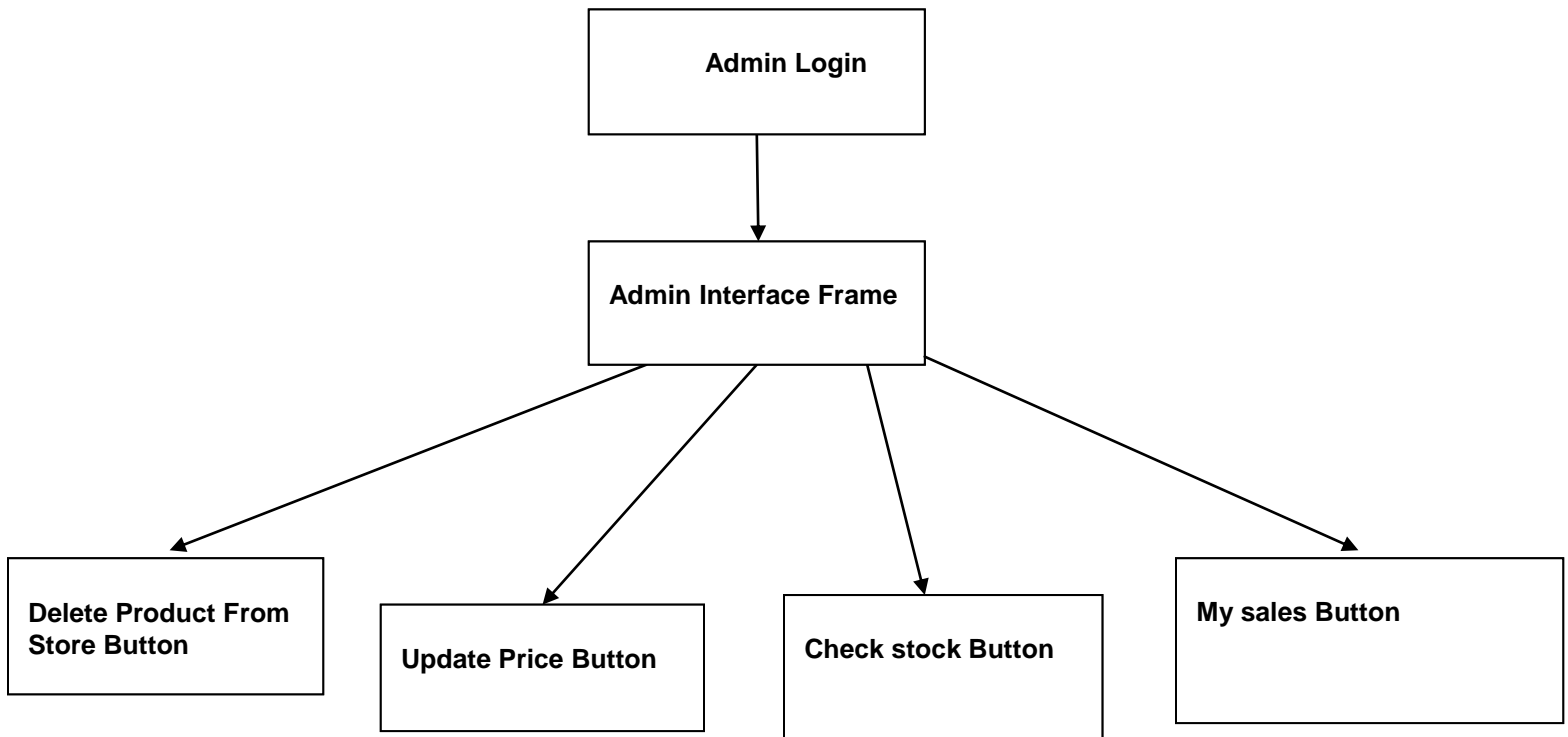
*The system consists of two main users, the Administrator and the customer. Both users select the relevant login page from the home page. There is also a separate registration page for the customer. The first page viewed by the user.*

### 2.3.2 Customer Register Page Frame

*There are multiple input fields in the customer registration frame to enter the customer's information. After the customer enters the information and presses the "Kayıt ol" button, the registration process is completed.*

### 2.3.3 Admin Interface Page Frame

*In the admin interface window, there are a total of 5 buttons: Add product to the store, Delete product from the store, Update price, Check stock, My sales and Logout.*



#### 2.3.3.1 Add Product to Store Button Function

*After the program asks the customer to enter the product name, it continues to ask the product size, product price, product color, and product stock quantity until the cancel button is pressed. After the Cancel button is pressed, the program asks the customer whether a new product will be added or not.*

### 2.3.3.2 Delete Product From Store Button Function

*A window showing the products in the store opens and the customer deletes the product by clicking on the product in the desired button format.*

### 2.3.3.3 Price Update Button Function

*When the button is pressed, the products in the store are listed in front of the customer and the customer is asked to enter the new desired price of the product he clicked. After logging in, the transaction is completed successfully.*

### 2.3.3.4 Check Stock Button Function

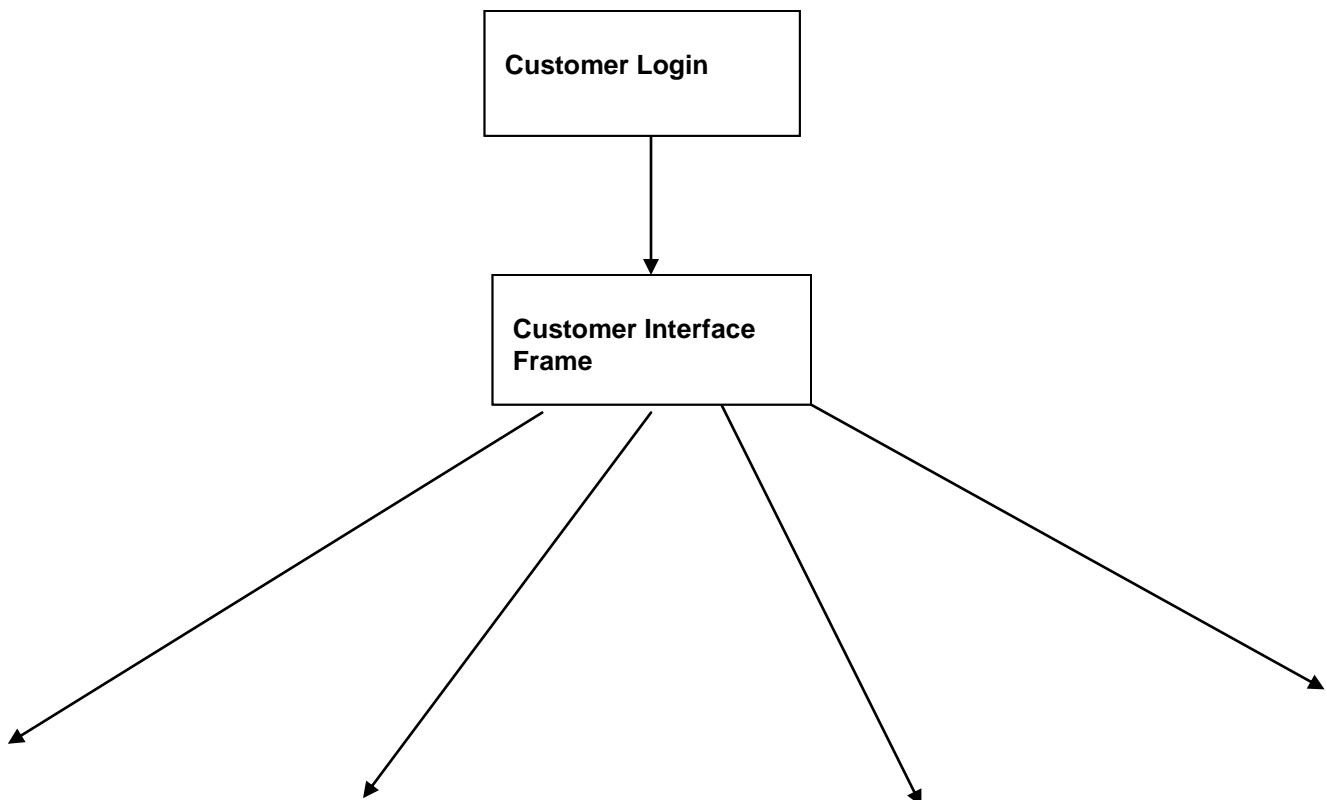
*The Frame that opens when the button is pressed shows the current stock information in the store.*

### 2.3.3.5 My Sales Button Function

*When the button is pressed, the purchased products are displayed with the purchaser's information.*

### 2.3.4 Customer Interface Page Frame

*In the customer interface window, there are a total of 5 buttons, Add product to cart, buy product, my purchases, delete product from cart and Logout.*



**Add Product to Cart**

**Buy Product**

**My Purchases**

**Delete Product From  
Cart**

#### 2.3.4.1 Add Product to Cart Button Function

*When the button is pressed, a window opens asking the customer how many products they want to add to the cart, and after the customer enters the desired value, the product is added to the cart and is also deducted from the store stock.*

#### 2.3.4.2 Buy Product Button Function

*The customer purchases the desired product among the products added to the cart by paying for it and entering its address.*

#### 2.3.4.3 My Purchases Button Function

*displays the products purchased when the customer presses the button*

#### 2.3.4.4 Delete Product From Cart Button Function

*When the button is pressed, the products in the basket are listed in front of the customer and the product that the customer clicks on is deleted from the basket.*

### 2.4 CONSTRAINTS AND ASSUMPTIONS

Since I do not create a session with a different ID for each customer in the program, when any customer logs out, the information in the txt I created for "My Purchases" is automatically deleted. Because when another user logs in, he can see the product in the "My Purchases" section even though he did not purchase it.

## 3. INTERFACES AND DATA STORAGE

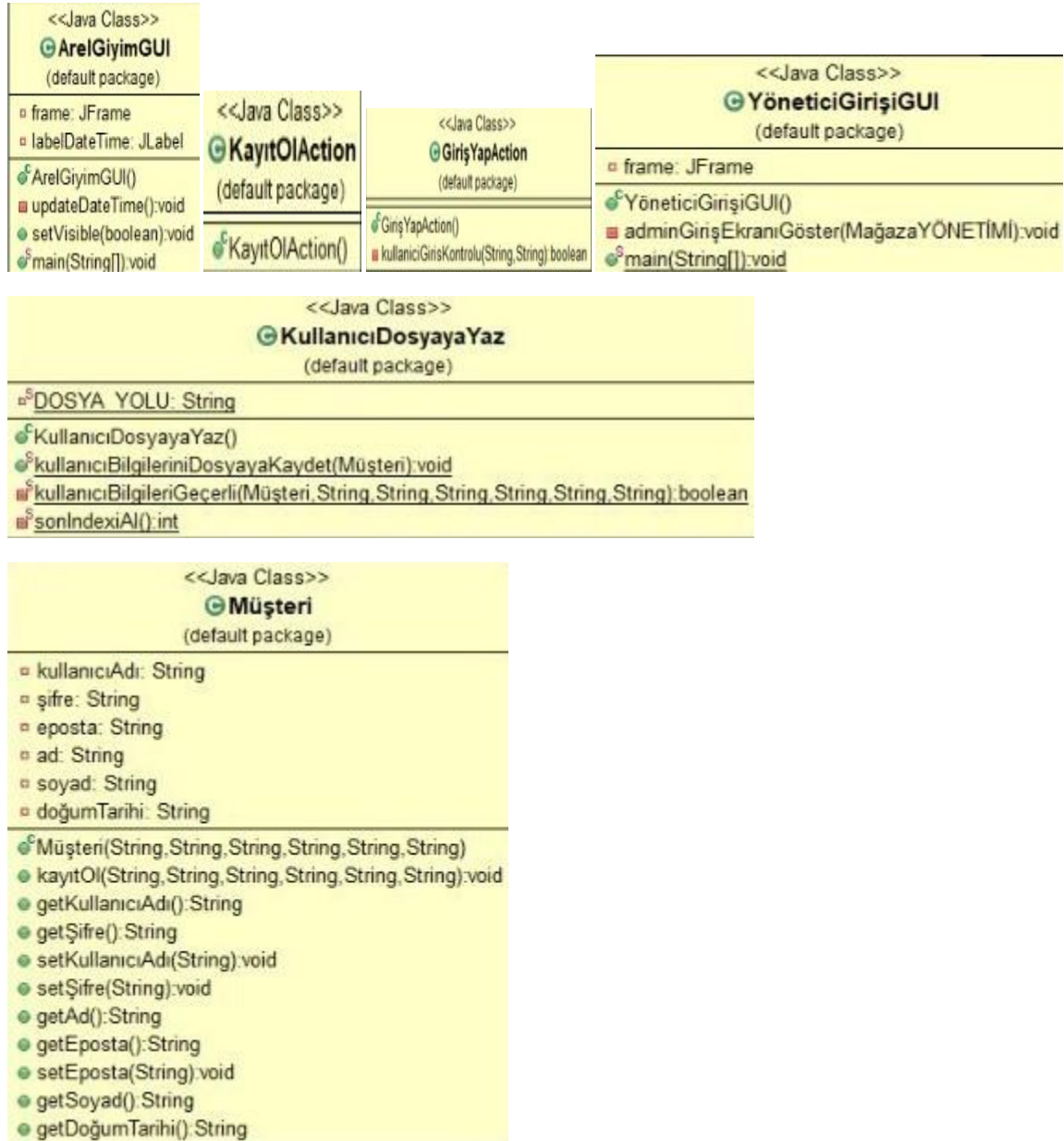
### 3.1 DESIGN EXPLANATION AND RATIONALE

The main concept and the structure type of the program is OOP. Which means the program is working with objects only. In the database management, due to the benefits of OOP, system does not need to create a file for each field. Another usage of OOP is the polymorphism and abstract

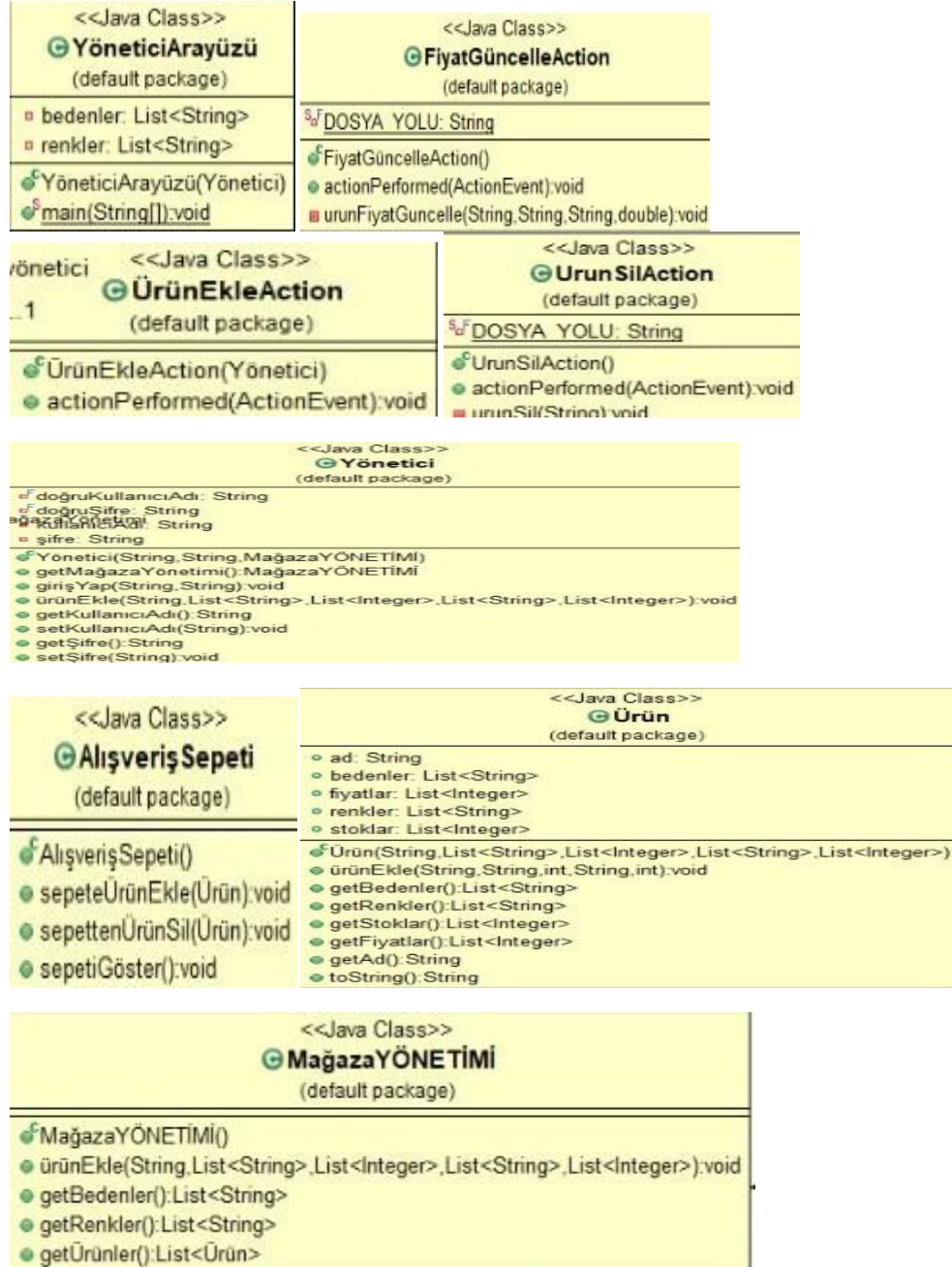
structures. It makes the program more easy to handle, readable and compatible with the ObjectOutputStream operations.

### 3.2 CLASS DIAGRAM

The ArelGiyimGUI and Login-Register Classes and Actions



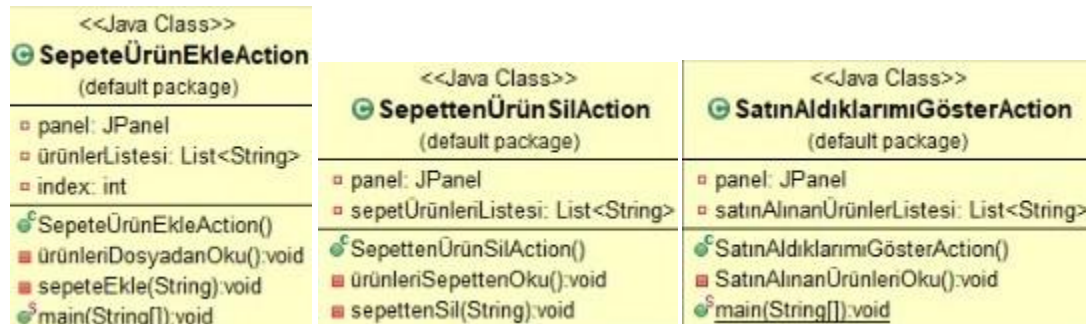
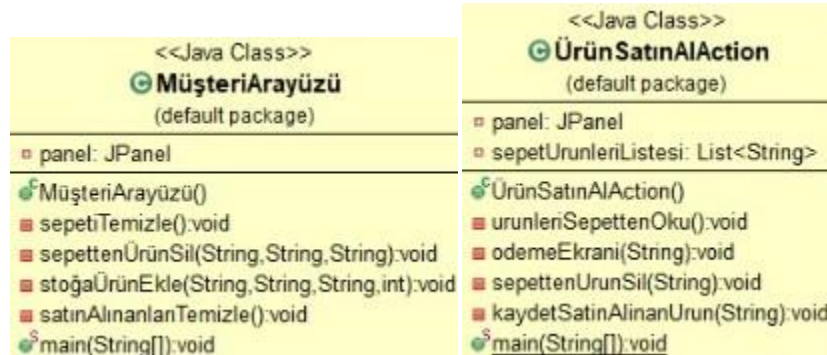
## Admin Interface (Yönetici Arayüzü) Classes and Frames

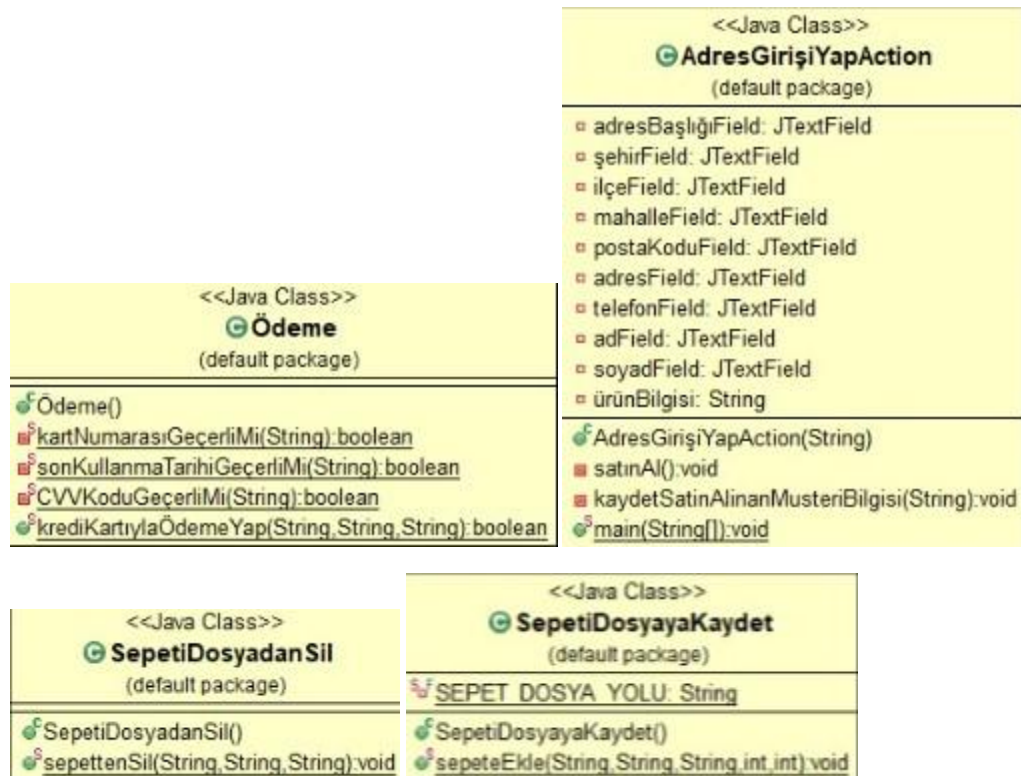






## Customer Interface (Müşteri Arayüzü) Classes and Frames





### 3.3 CLASS DESCRIPTION

### 3.4 CLASSES IN AREL GİYİM MANAGEMENT SYSTEM

#### ArelGiyimGUI Class

##### Attributes

- **private JFrame frame:** The main window of the application.
- **private JLabel labelDateTime:** Label to display the current date and time.

##### Methods

- **ArelGiyimGUI():**
  - Constructor method that sets up the user interface components.
  - Initializes the main frame and sets its properties.
  - Creates a panel with a grid layout to hold the components.
  - Adds the **labelDateTime** to the panel and calls **updateDateTime()** to set the initial date and time.
  - Adds a JLabel with the text "AREL GİYİM" styled as a title.

- Starts a timer that updates the **labelDateTime** every second.
- Adds buttons for "Yönetici Girişi" (Admin Login), "Müşteri Kayıt Ol" (Customer Register), and "Müşteri Girişi" (Customer Login) with their respective action listeners.
- **private void updateDateTime():**
  - Method that updates the **labelDateTime** with the current date and time.
  - Uses **SimpleDateFormat** to format the date and time.
- **public void setVisible(boolean visible):**
  - Sets the visibility of the main frame.
- **public static void main(String[] args):**
  - The main method that runs the application.
  - Uses **SwingUtilities.invokeLater** to ensure the GUI is created and updated on the Event Dispatch Thread.

#### Action Listeners (Inner Classes)

- **ActionListener for Timer:**
  - Updates the **labelDateTime** every second.
- **ActionListener for btnYöneticiGirişi (Admin Login Button):**
  - Disposes the current frame and opens a new **YöneticiGirişiGUI** window.
- **ActionListener for btnMüşteriKayıtOl (Customer Register Button):**
  - Disposes the current frame and opens a new **KayıtOlAction** window.
- **ActionListener for btnMüşteriGirişi (Customer Login Button):**
  - Disposes the current frame and opens a new **GirişYapAction** window.

## KayıtOlAction Class

#### Attributes

- **JLabel lblKullanıcıAdı:** Label for "Username".
- **TextField txtKullanıcıAdı:** Text field for entering the username.
- **JLabel lblŞifre:** Label for "Password".
- **JPasswordField txtŞifre:** Password field for entering the password.

- **JLabel lblEposta:** Label for "Email".
- **TextField txtEposta:** Text field for entering the email address.
- **JLabel lblAd:** Label for "First Name".
- **TextField txtAd:** Text field for entering the first name.
- **JLabel lblSoyad:** Label for "Last Name".
- **TextField txtSoyad:** Text field for entering the last name.
- **JLabel lblDoğumTarih:** Label for "Date of Birth".
- **TextField txtDoğumTarih:** Text field for entering the date of birth.
- **Button btnKayıtOl:** Button for "Register".
- **Button geriButton:** Button for "Back".

## Methods

- **KayıtOlAction():**
  - Constructor method that sets up the user interface components.
  - Initializes the frame, sets its title, default close operation, size, and location.
  - Creates a panel with a grid layout to hold the form components (labels and text fields).
  - Adds the labels and text fields to the panel.
  - Adds an action listener to the "Register" button to handle user registration.
    - Collects the entered data.
    - Creates a **Müşteri** object with the collected data.
    - Saves the customer information to a file using **KullanıcıDosyayaYaz.kullanıcıBilgileriniDosyayaKaydet(müşteri)**.
    - Opens the **ArelGiyimGUI** and disposes of the current frame.
  - Adds an action listener to the "Back" button to return to the main screen.
    - Disposes of the current frame and opens the **ArelGiyimGUI**.
  - Adds the "Back" button to a bottom panel with a flow layout.
  - Sets the visibility of the frame to true.

## GirişYapAction Class

### Attributes

- **JLabel lblKullanıcıAdı**: Label for "Username".
- **TextField txtKullanıcıAdı**: Text field for entering the username.
- **JLabel lblŞifre**: Label for "Password".
- **JPasswordField txtŞifre**: Password field for entering the password.
- **JButton btnGirişYap**: Button for "Login".
- **JButton geriButton**: Button for "Back".

### Methods

- **GirişYapAction()**:
  - Constructor method that sets up the user interface components.
  - Initializes the frame, sets its title, default close operation, size, and location.
  - Creates a panel with a grid layout to hold the form components (labels and text fields).
  - Adds the labels and text fields to the panel.
  - Adds an action listener to the "Login" button to handle user login.
    - Retrieves the entered username and password.
    - Calls the **kullaniciGirisKontrolu** method to verify the username and password.
    - If the credentials are correct, opens the **MüşteriArayüzü** and disposes of the current frame.
    - If the credentials are incorrect, displays an error message dialog.
  - Adds an action listener to the "Back" button to return to the main screen.
    - Disposes of the current frame and opens the **ArelGiyimGUI**.
  - Adds the "Back" button to a bottom panel with a flow layout.
  - Sets the visibility of the frame to true.
- **private boolean kullaniciGirisKontrolu(String kullanıcıAdı, String şifre)**:
  - Checks if the provided username and password match any entry in the file **kullanıcılar.txt**.
  - Reads the file line by line, splits each line by comma, and compares the username and password.

- Returns **true** if a match is found, otherwise returns **false**.

## **YöneticiGirişGUI Class**

### **Attributes**

- **private JFrame frame:** The main frame of the admin login GUI.

### **Methods**

- **YöneticiGirişGUI():**
  - Constructor method that initializes the main frame and sets its title, close operation, size, and location.
  - Creates an instance of **MağazaYÖNETİMİ**.
  - Calls the **adminGirişEkranıGöster** method to display the admin login screen.
- **private void adminGirişEkranıGöster(MağazaYÖNETİMİ mağazaYönetimi):**
  - Sets up the admin login screen with a panel using BorderLayout.
  - Adds a title label "Yönetici Girişi" at the top.
  - Creates a grid layout panel for login fields (username and password) and adds them to the center.
  - Adds action listeners to handle login button actions:
    - On clicking "Giriş Yap", checks if the entered username and password match predefined values.
    - If valid, disposes of the current frame and opens the **YöneticiArayüzü**.
    - If invalid, shows an error message dialog.
  - Adds a "Back" button to return to the main screen:
    - On clicking "Geri", disposes of the current frame and opens the **ArelGiyimGUI**.
  - Sets the visibility of the frame to true.
- **public static void main(String[] args):**
  - Main method that uses **SwingUtilities.invokeLater** to run the **YöneticiGirişGUI**.

## KullanıcıDosyayaYaz Class

### Attributes

- **private static String DOSYA\_YOLU:**
  - The file path where the user information is stored.

### Methods

- **public static void kullanıcıBilgileriniDosyayaKaydet(Müşteri müşteri):**
  - Saves the customer information to the file if the user information is valid.
  - Uses a BufferedWriter to append the new customer information to the file.
- **private static boolean kullanıcıBilgileriGeçerli(Müşteri müşteri, String kullanıcıAdı, String şifre, String ad, String soyad, String eposta, String doğumTarihi):**
  - Validates the customer information to ensure it matches the input fields.
- **private static int sonIndexAl():**
  - Reads the file to get the last index used for the customer records.
  - Parses the last line of the file to determine the last used index.

## Müşteri Class

### Attributes

- **private String kullanıcıAdı:**
  - The username of the customer.
- **private String şifre:**
  - The password of the customer.
- **private String eposta:**
  - The email of the customer.
- **private String ad:**
  - The first name of the customer.
- **private String soyad:**
  - The last name of the customer.
- **private String doğumTarihi:**
  - The birth date of the customer.

## Constructor

- **public Müşteri(String kullanıcıAdı, String şifre, String eposta, String ad, String soyad, String doğumTarihi):**
  - Initializes a new customer with the given username, password, email, first name, last name, and birth date.

## Methods

- **public void kayıtOl(String kullanıcıAdı, String şifre, String eposta, String ad, String soyad, String doğumTarihi):**
  - Sets the customer's username, password, email, first name, last name, and birth date.
  - Prints a message indicating the new customer has been registered.
- **public String getKullanıcıAdı():**
  - Returns the username of the customer.
- **public String getŞifre():**
  - Returns the password of the customer.
- **public void setKullanıcıAdı(String kullanıcıAdı):**
  - Sets the username of the customer.
- **public void setŞifre(String şifre):**
  - Sets the password of the customer.
- **public String getAd():**
  - Returns the first name of the customer.
- **public String getEposta():**
  - Returns the email of the customer.
- **public void setEposta(String eposta):**
  - Sets the email of the customer.
- **public String getSoyad():**
  - Returns the last name of the customer.
- **public String getDoğumTarihi():**
  - Returns the birth date of the customer.



## YöneticiArayüzü Class

### Attributes

- **private Yönetici yönetici:**
  - The administrator object associated with the interface.
- **private List<String> bedenler:**
  - A list of sizes available in the store.
- **private List<String> renkler:**
  - A list of colors available in the store.

### Constructor

- **public YöneticiArayüzü(Yönetici yönetici):**
  - Initializes the admin interface with the given admin.
  - Sets up the JFrame with title, size, and default close operation.
  - Retrieves the list of sizes and colors from the store management.
  - Adds components and action listeners for different functionalities.

### Methods

- **public static void main(String[] args):**
  - Main method to run the application.
  - Creates a **MağazaYÖNETİMİ** and **Yönetici** object.
  - Initializes the **YöneticiArayüzü** with the created admin.

### GUI Components and Action Listeners

- **JButton ürünEkleButon:**
  - Button to add products to the store.
  - Action listener: **ÜrünEkleAction(yönetici)** to handle product addition.
- **JButton ürünSilButon:**
  - Button to remove products from the store.
  - Action listener: **UrunSilAction()** to handle product removal.
- **JButton fiyatGüncelleButon:**
  - Button to update product prices.

- Action listener: Opens the price update interface using **FiyatGüncelleAction(yönetici)**.
- **JButton stokKontrolButon:**
  - Button to check stock levels.
  - Action listener: **StokKontrolAction()** to handle stock checking.
- **JButton satılanlarıGörüntüleButon:**
  - Button to view sales.
  - Action listener: **SatışlarıGörüntüleAction()** to handle sales viewing.
- **JButton çıkışButon:**
  - Button to log out.
  - Action listener: Shows a logout message and returns to the main GUI (**ArelGiyimGUI**).

## **FiyatGüncelleAction Class**

### **Attributes**

- **private Yönetici yönetici:**
  - The administrator associated with the action.
- **private static final String DOSYA\_YOLU:**
  - The file path where the product data is stored.

### **Constructor**

- **public FiyatGüncelleAction(Yönetici yönetici):**
  - Initializes the action with the given administrator.

### **Methods**

- **public void actionPerformed(ActionEvent e):**
  - Defines the action to be performed when the event is triggered.
  - Displays a frame with a list of products and their details, allowing price updates.
- **private void urunFiyatGuncelle(String ürünAdı, String beden, String renk, double yeniFiyat):**
  - Updates the price of a specified product in the file.

- Reads the file, updates the product's price, and writes the updated data back to the file.

## **ÜrünEkleAction Class**

### **Attributes**

- **private Yönetici yönetici:**
  - The administrator who will add the products.

### **Constructor**

- **public ÜrünEkleAction(Yönetici yönetici):**
  - Initializes the action with the given administrator.

### **Methods**

- **public void actionPerformed(ActionEvent e):**
  - Defines the action to be performed when the event is triggered.
  - Opens dialogs to input product details (name, sizes, prices, colors, and stock quantities).
  - Allows multiple products to be added in a loop until the user decides to stop.
  - Updates the administrator's product list and writes the product details to a file.

## **UrunSilAction Class**

### **Attributes**

- **private static final String DOSYA\_YOLU:**
  - The path to the file containing product information.

### **Constructor**

- **N/A:**
  - This class does not require a constructor since it implements ActionListener and performs actions directly within the actionPerformed method.

### **Methods**

- **public void actionPerformed(ActionEvent e):**
  - Displays a JFrame to list all products read from a file with buttons to delete each product.

- Sets up the JFrame with a large size and centers it on the screen.
- Uses a GridLayout for the panel that will hold buttons for each product.
- Reads the products from the file, creating a delete button for each product with its details.
- Adds the delete buttons to the panel and the panel to a JScrollPane for scrolling.
- Shows the frame.
- **private void urunSil(String urunAdi):**
  - Deletes a product with the given name from the file.
  - Reads all lines from the file, skips the line with the matching product name, and writes the remaining lines back to the file.
  - Uses BufferedReader to read the file and BufferedWriter to rewrite it with updated content.

## Yönetici Class

### Attributes

- **private final String doğruKullanıcıAdı:**
  - The correct username for the admin. Set to "HüseyinKaya".
- **private final String doğruŞifre:**
  - The correct password for the admin. Set to "HüseyinKaya123".
- **private String kullanıcıAdı:**
  - The username entered by the admin.
- **private String şifre:**
  - The password entered by the admin.
- **public MağazaYÖNETİMİ mağazaYönetimi:**
  - An instance of the MağazaYÖNETİMİ class, representing the store management system.

### Constructor

- **public Yönetici(String kullanıcıAdı, String şifre, MağazaYÖNETİMİ mağazaYönetimi):**
  - Initializes the Yönetici object with the given username, password, and store management system.

## Methods

- **public MağazaYÖNETİMİ getMağazaYönetimi():**
  - Returns the instance of MağazaYÖNETİMİ associated with this admin.
- **public void girişYap(String kullanıcıAdı, String şifre):**
  - Checks if the provided username and password match the correct admin credentials.
  - Prints a success message if the credentials are correct.
- **public void ürünEkle(String ürünAdı, List<String> bedenler, List<Integer> fiyatlar, List<String> renkler, List<Integer> stoklar):**
  - Calls the ürünEkle method of the MağazaYÖNETİMİ instance to add a new product with the given details.
- **public String getKullanıcıAdı():**
  - Returns the username of the admin.
- **public void setKullanıcıAdı(String kullanıcıAdı):**
  - Sets the username of the admin.
- **public String getŞifre():**
  - Returns the password of the admin.
- **public void setŞifre(String şifre):**
  - Sets the password of the admin.

## AlışverişSepeti Class

### Attributes

- **private List<Ürün> sepettekiÜrünler:**
  - A list that holds the products (instances of the Ürün class) in the shopping cart.

### Constructor

- **public AlışverişSepeti():**
  - Initializes an empty list to store products in the shopping cart.

### Methods

- **public void sepeteÜrünEkle(Ürün ürün):**

- Adds the specified product to the shopping cart.
- Prints a message indicating that the product has been added.
- **public void sepettenÜrünSil(Ürün ürün):**
  - Removes the specified product from the shopping cart if it is present.
  - Prints a message indicating that the product has been removed.
  - If the product is not in the cart, prints a message indicating that the product was not found in the cart.
- **public void sepetiGöster():**
  - Displays the products currently in the shopping cart.
  - If the cart is empty, prints a message indicating that the cart is empty.
  - Otherwise, prints the names of all products in the cart.

## Ürün Class

### Attributes

- **public String ad:**
  - The name of the product.
- **public List<String> bedenler:**
  - A list of available sizes for the product.
- **public List<Integer> fiyatlar:**
  - A list of prices for the product.
- **public List<String> renkler:**
  - A list of available colors for the product.
- **public List<Integer> stoklar:**
  - A list of stock quantities for the product.

### Constructor

- **public Ürün(String ad, List<String> bedenler, List<Integer> fiyatlar, List<String> renkler, List<Integer> stoklar):**
  - Initializes the product with its name, sizes, prices, colors, and stock quantities.

### Methods

- **public void ürünEkle(String ad, String beden, int fiyat, String renk, int stok):**

- Adds a new size, price, color, and stock quantity to the existing lists for the product.
- **public List<String> getBedenler():**
  - Returns the list of available sizes for the product.
- **public List<String> getRenkler():**
  - Returns the list of available colors for the product.
- **public List<Integer> getStoklar():**
  - Returns the list of stock quantities for the product.
- **public List<Integer> getFiyatlar():**
  - Returns the list of prices for the product.
- **public String getAd():**
  - Returns the name of the product.
- **@Override public String toString():**
  - Returns a string representation of the product, including its name and the first color in the list.

## **MağazaYÖNETİMİ Class**

### **Attributes**

- **public List<Ürün> ürünler:**
  - A list to store the products available in the store.

### **Constructor**

- **public MağazaYÖNETİMİ():**
  - Initializes the product list as an empty list.

### **Methods**

- **public void ürünEkle(String ürünAdı, List<String> bedenler, List<Integer> fiyatlar, List<String> renkler, List<Integer> stoklar):**
  - Adds a new product to the store with the given name, sizes, prices, colors, and stock quantities.
  - Creates a new **Ürün** instance and adds it to the **ürünler** list.
  - Prints a message indicating that the product has been added to the store.
- **public List<String> getBedenler():**

- Returns a list of all sizes available in the store.
- Iterates through the **ürünler** list and collects all sizes from each product.
- **public List<String> getRenkler():**
  - Returns a list of all colors available in the store.
  - Iterates through the **ürünler** list and collects all colors from each product.
- **public List<Ürün> getÜrünler():**
  - Returns the list of products available in the store.

## **SatışlarıGörüntüleAction Class**

### **Attributes**

- **private static final String DOSYA\_YOLU:** Path to the file containing sales information.

### **Methods**

- **public void actionPerformed(ActionEvent e):**
  - Implements the action to be performed when the "Satışları Görüntüle" button is clicked.
  - Creates a new frame to display sales information.
  - Reads sales information from the file specified by **DOSYA\_YOLU**.
  - Parses each line of sales information, splits it into parts, and constructs a panel for each sale.
  - Each panel contains labels displaying customer information, purchased product information, and address information.
  - Adds spacing and separators between panels for better visualization.
  - Adds each panel to the main panel.
  - Wraps the main panel in a scroll pane for scrolling if necessary.
  - Displays the frame.

## **StokKontrolAction Class**

### **Attributes**

- **private static final String DOSYA\_YOLU:** Path to the file containing product information.

### **Methods**



- **public void actionPerformed(ActionEvent e):**
  - Implements the action to be performed when the "Stok Kontrolü" button is clicked.
  - Creates a new frame to display stock information.
  - Reads product information from the file specified by **DOSYA\_YOLU**.
  - Parses each line of product information, splits it into parts, and constructs a panel for each product.
  - Each panel contains labels displaying product name, size, color, and stock quantity.
  - Adds spacing between panels for better visualization.
  - Adds each panel to the frame.
  - Displays the frame.

## ÜrünleriDosyayaYaz Class

### Attributes

- **private static final String DOSYA\_YOLU:** Path to the file where product information will be written.

### Methods

- **public static void ürünleriYaz(String ürünAdı, List<String> bedenler, List<String> renkler, List<Integer> stoklar, List<Integer> fiyatlar):**
  - Writes product information to the file specified by **DOSYA\_YOLU**.
  - Appends the new product information to the end of the file.
  - Retrieves the last index from the file using the **getSonIndex** method to determine the index of the new product entry.
  - Iterates over the lists of sizes, colors, stocks, and prices, writing each product entry line by line to the file.
- **private static int getSonIndex():**
  - Reads the lines from the file specified by **DOSYA\_YOLU**.
  - Parses the last line to extract the index of the last product entry.
  - Returns the last index.

## ÜrünleriDosyadanSil Class

### Attributes

- **private static final String DOSYA\_YOLU:** Path to the file where product information is stored.

### Methods

- **public static void ürünSil(String ürünAdı):**
  - Reads all lines from the file specified by **DOSYA\_YOLU** using **Files.readAllLines** method.
  - Filters out the lines containing the given product name using a stream and **filter** operation, creating a list of lines without the specified product.
  - Writes the updated lines (without the removed product) back to the file, effectively removing the product entry.
  - Handles **IOException** if any error occurs during file reading or writing.

## MüşteriArayüzü Class

### Attributes

- **panel:** A **JPanel** used to organize and display the buttons in a grid layout.

### Constructor

- **MüşteriArayüzü():** Sets up the frame and panel, adds buttons for various actions, and makes the frame visible.

### Methods

- **private void sepetiTemizle():** Reads the products from the file where the products in the shopping cart are stored, adds them back to the inventory, and removes them from the shopping cart file.
- **private void sepettenÜrünSil(String ürünAdı, String beden, String renk):** Removes a specific product from the shopping cart file based on its name, size, and color.
- **private void stoğaÜrünEkle(String ürünAdı, String beden, String renk, int adet):** Adds a specified number of a product back to the inventory after removing them from the shopping cart.
- **private void satınAlınanlarıTemizle():** Clears the file where the purchased products are stored.
- **public static void main(String[] args):** Launches the application by creating an instance of **MüşteriArayüzü** and setting it visible.

## ÜrünSatınAlAction Class

### Attributes

- **panel**: A **JPanel** to organize and display the products available for purchase.
- **sepetUrunleriListesi**: A list to store the products in the shopping cart.

### Constructor

- **ÜrünSatınAlAction()**: Sets up the frame, reads the products from the shopping cart file, creates buttons for each product, and adds action listeners to handle the purchase process.

### Methods

- **private void urunleriSepettenOku()**: Reads the products from the shopping cart file and populates the **sepetUrunleriListesi**.
- **private void odemeEkrani(String urun)**: Displays the payment screen where users can enter their payment details. Upon successful payment, it removes the purchased product from the shopping cart and saves it as a purchased item.
- **private void sepettenUrunSil(String urun)**: Removes the purchased product from the shopping cart file.
- **private void kaydetSatinAlinanUrun(String urun)**: Saves the purchased product to the file where purchased items are stored.
- **public static void main(String[] args)**: Instantiates the **ÜrünSatınAlAction** class to launch the purchase interface.

## SepeteÜrünEkleAction Class

### Attributes

- **panel**: A **JPanel** to organize and display the products available for adding to the cart.
- **ürünlerListesi**: A list to store the products read from the file.
- **index**: An integer variable to keep track of the index while iterating through the products.

### Constructor

- **SepeteÜrünEkleAction()**: Sets up the frame, reads the products from the file, creates buttons for each product, and adds action listeners to handle adding products to the cart.

### Methods

- **private void ürünleriDosyadanOku()**: Reads the products from the file and populates the **ürünlerListesi**.
- **private void sepeteEkle(String ürün)**: Handles the process of adding a product to the cart. It prompts the user for the quantity they want to add, checks if there is enough stock, updates the stock quantity in the file, and adds the product to the cart file.

## SepettenÜrünSilAction Class

### Attributes

- **panel:** A **JPanel** to organize and display the products in the cart for removal.
- **sepetÜrünleriListesi:** A list to store the products in the shopping cart.

### Constructor

- **SepettenÜrünSilAction():** Sets up the frame, reads the products from the cart file, creates buttons for each product in the cart, and adds action listeners to handle removing products from the cart.

### Methods

- **private void ürünleriSepettenOku():** Reads the products from the cart file and populates the **sepetÜrünleriListesi**.
- **private void sepettenSil(String ürün):** Handles the process of removing a product from the cart. It removes the product from the cart file and updates the stock quantity of the corresponding product in the products file.

## SatınAldıklarımıGösterAction Class

### Attributes

- **panel:** A **JPanel** to organize the UI components.
- **textArea:** A **JTextArea** to display the purchased items.
- **satınAlınanÜrünlerListesi:** A list to store the purchased items.

### Constructor

- **SatınAldıklarımıGösterAction():** Sets up the frame, initializes the UI components, reads the purchased items from the file, formats the purchased items, and displays them in the text area.

### Methods

- **private void satınAlınanÜrünleriOku():** Reads the purchased items from the file and populates the **satınAlınanÜrünlerListesi**.
- **public static void main(String[] args):** Entry point of the application. It creates an instance of **SatınAldıklarımıGösterAction** to display the purchased items.

## Ödeme Class

### Methods

- **private static boolean kartNumarasıGeçerliMi(String kartNumarası):** Validates the credit card number format. It checks if the credit card number consists of 16 digits.
- **private static boolean sonKullanmaTarihiGeçerliMi(String sonKullanmaTarihi):** Validates the expiration date format. It checks if the date is in the format MM/YY.
- **private static boolean CVVKoduGeçerliMi(String cvv):** Validates the CVV (Card Verification Value) code format. It checks if the CVV consists of 3 digits.
- **public static boolean krediKartıylaÖdemeYap(String kartNumarası, String sonKullanmaTarihi, String cvv):** Performs the payment process with a credit card. It validates the credit card number, expiration date, and CVV code using the helper methods. If all validations pass, it prints "Ödeme başarıyla tamamlandı." (Payment completed successfully) and returns **true**. Otherwise, it prints the corresponding error message and returns **false**.

## AdresGirişiYapAction Class

### Fields

- **adresBaşlığıField:** JTextField for entering the address title.
- **şehirField:** JTextField for entering the city.
- **ilçeField:** JTextField for entering the district.
- **mahalleField:** JTextField for entering the neighborhood.
- **postaKoduField:** JTextField for entering the postal code.
- **adresField:** JTextField for entering the full address.
- **telefonField:** JTextField for entering the phone number.
- **adField:** JTextField for entering the first name.
- **soyadField:** JTextField for entering the last name.
- **ürünBilgisi:** String variable to store the product information.

### Constructor

- **AdresGirişiYapAction(String ürün):** Initializes the JFrame with the title "Adres Bilgileri". It sets up the layout with 12 rows and 2 columns. It creates JLabels and JTextFields for entering customer information and address details. Additionally, it creates a "Satın Al" JButton with an ActionListener to trigger the purchase process when clicked.

### Methods

- **satınAl()**: Retrieves the input values from the text fields, constructs the complete purchase information string by concatenating the product information and customer address details, and saves this information to a file using the **kaydetSatinAlinanMusteriBilgisi()** method. It also performs validation to ensure that all required fields are filled. If successful, it displays a success message and closes the window.
- **kaydetSatinAlinanMusteriBilgisi(String bilgi)**: Writes the provided information to a file named "satinalanmusteribilgileri.txt" for record-keeping.

#### main Method

- **main(String[] args)**: Entry point of the program. It creates an instance of **AdresGirişiYapAction** with an empty string as the product information, effectively initializing the address input form.

## SepetiDosyadanSil Class

#### Method

- **sepettenSil(String urunAdi, String beden, String renk)**: This method takes the name of the product (**urunAdi**), its size (**beden**), and its color (**renk**) as parameters. It reads the contents of the file "sepettekiurunler.txt" line by line into a list (**ürünlerListesi**). Then, it removes the lines that match the provided product name, size, and color using the **removeIf** method along with a lambda expression. After removing the specified product, it writes the updated list back to the file, effectively updating the shopping cart.

## SepetiDosyayaKaydet Class

#### Constants

- **SEPET\_DOSYA\_YOLU**: This constant holds the file path where the shopping cart data will be stored.

#### Method

- **sepeteEkle(String ürünAdı, String beden, String renk, int adet, int fiyat)**: This method takes the name of the product (**ürünAdı**), its size (**beden**), its color (**renk**), quantity (**adet**), and price (**fiyat**) as parameters. It first reads all the existing lines from the shopping cart file into a list **mevcutÜrünler**. Then, it calculates the index for the new product based on the size of the existing list. After that, it opens a **BufferedWriter** to append data to the shopping cart file. It constructs the product information string (**ürünBilgisi**) by concatenating the index, product name, size, color, quantity, and price with commas. It then writes this information to the file as a new line and prints a success message to the console. If an exception occurs during the process, it prints an error message along with the exception details to the standard error stream.

## 4. DYNAMIC MODEL

### 4.1 SCENARIOS

*Clicks the "Yönetici Girişi" button to allow the administrator to log in.*



Arel Giyim Başlangıç Sayfası

26/05/2024 22:39:34

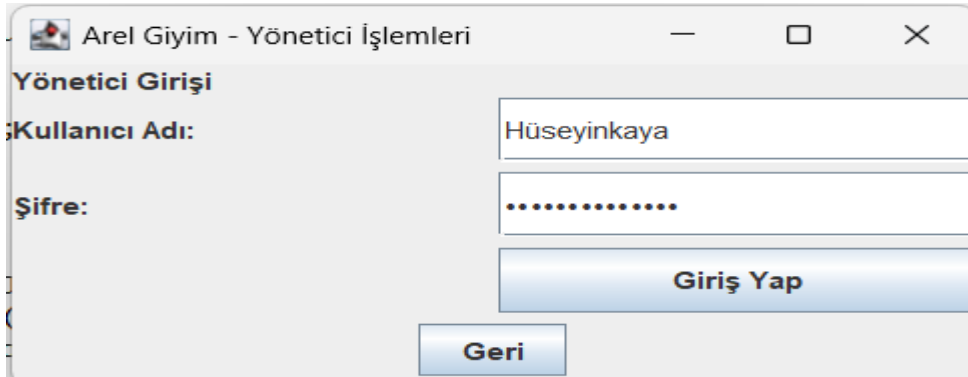
**AREL GİYİM**

Yönetici Girişi

Müşteri Kayıt Ol

Müşteri Girişi

*To log in, the administrator writes his information and presses the "Giriş Yap" button.*



Arel Giyim - Yönetici İşlemleri

**Yönetici Girişi**

Kullanıcı Adı: Hüseyinkaya

Şifre: .....

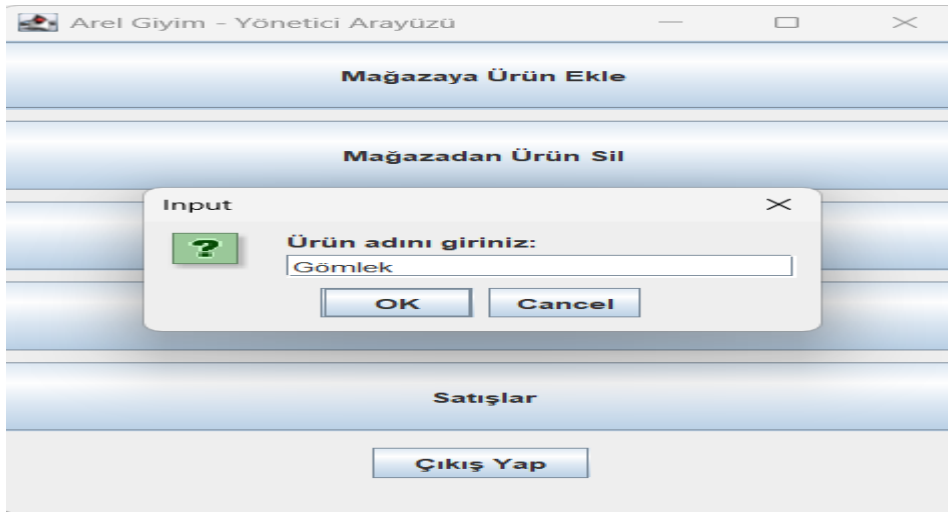
Giriş Yap

Geri

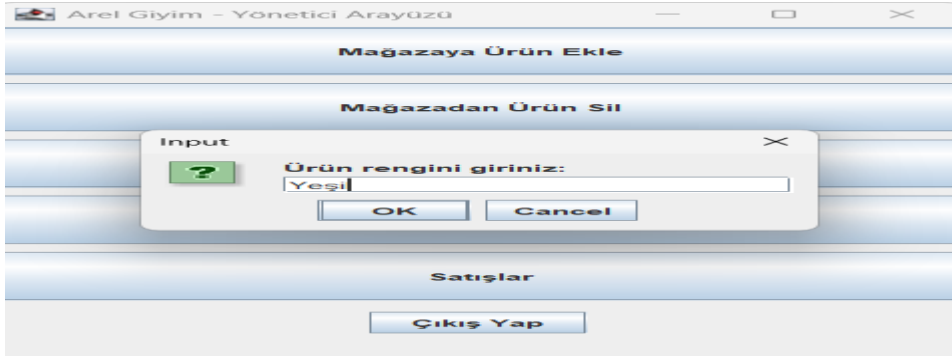
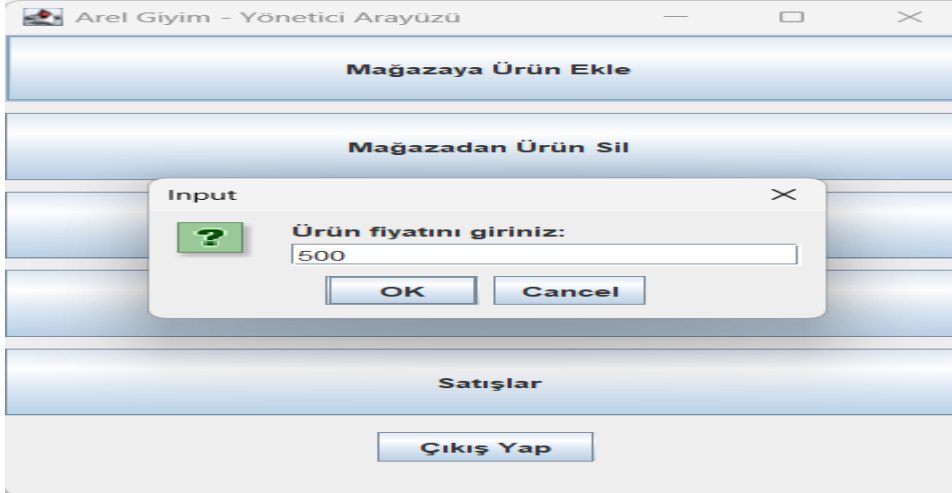
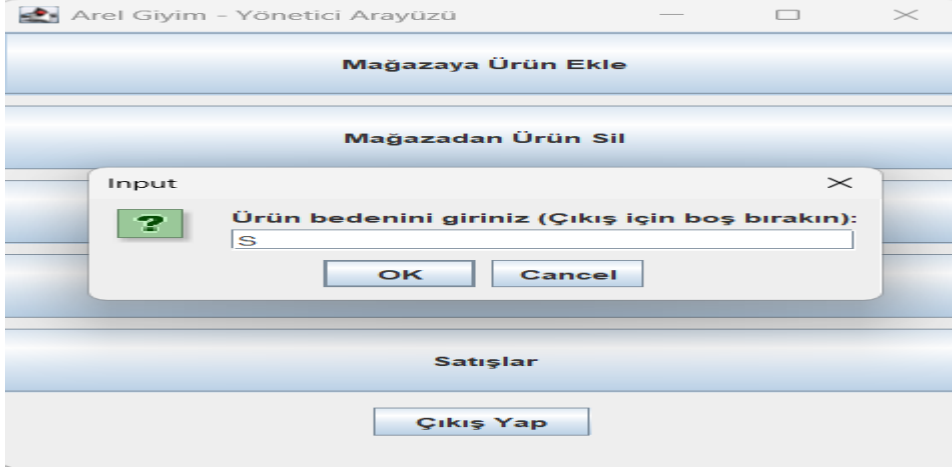
*login successful, Admin Interface screen opened*

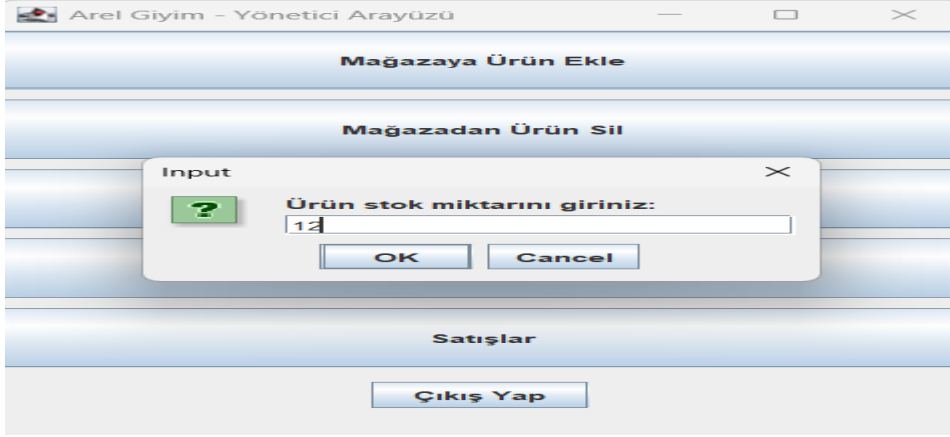


*The manager wanted to add a product to the store and pressed the "Mağazaya Ürün Ekle" button and entered all the information in order*

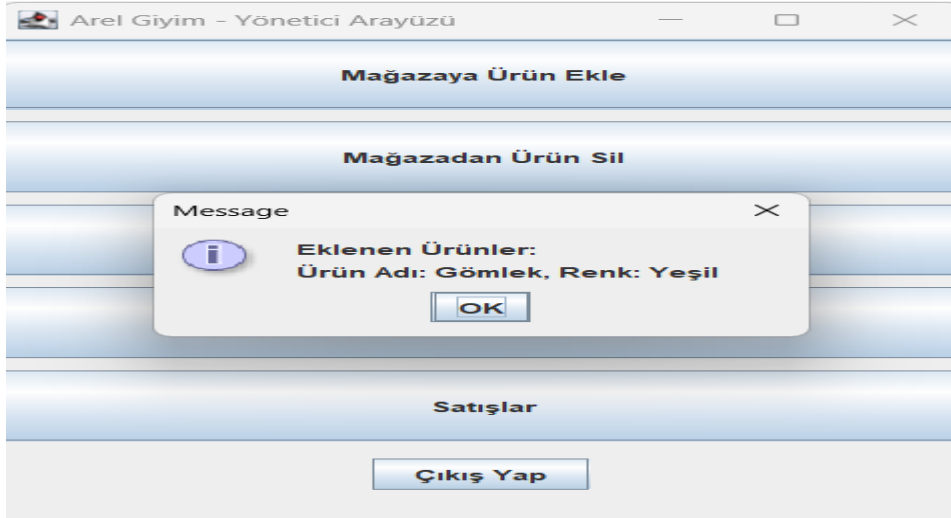
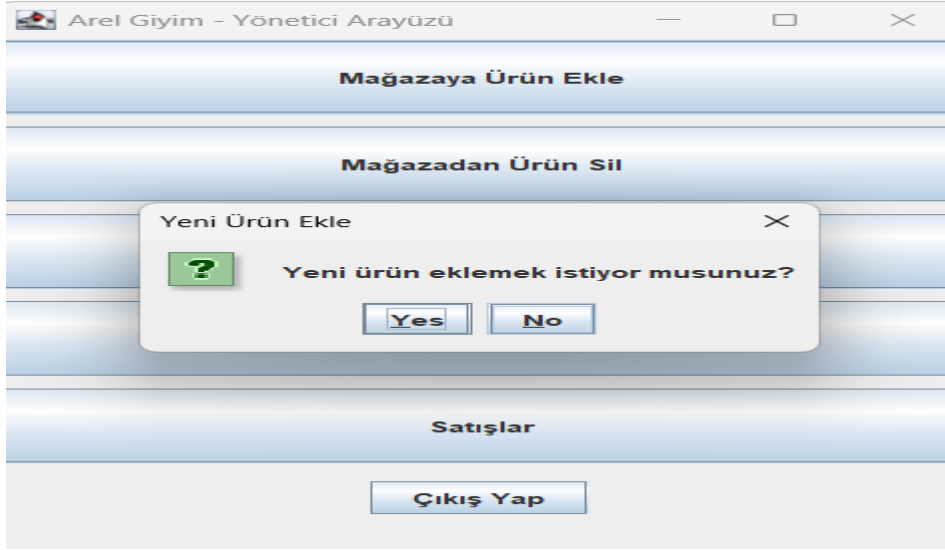








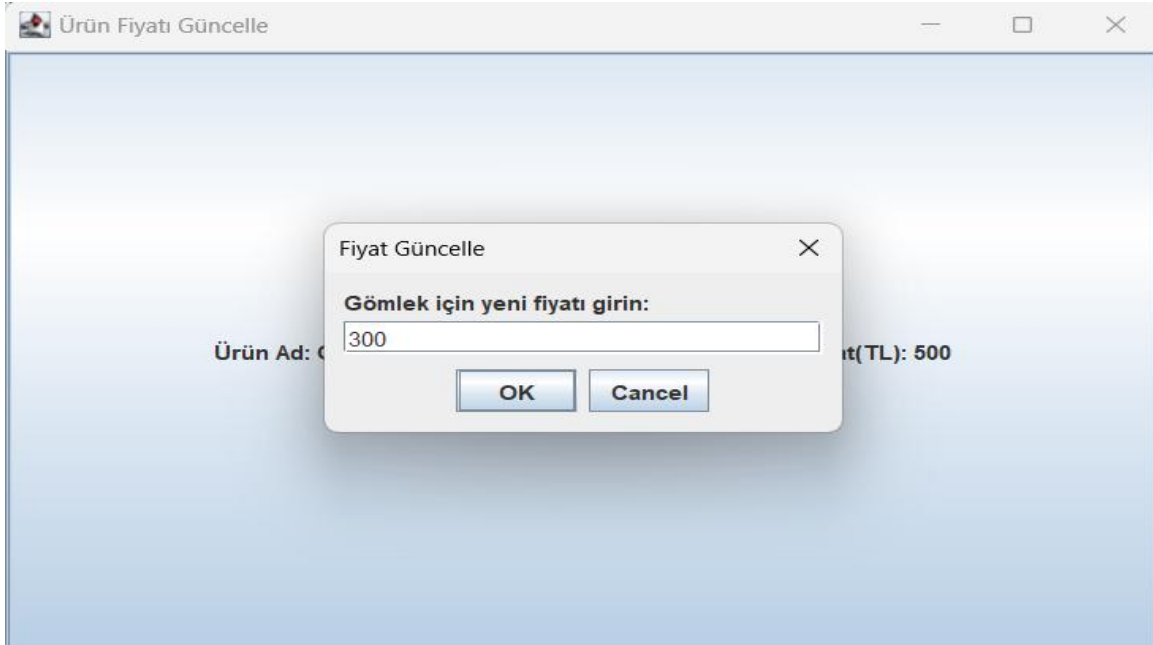
*Afterwards, the program asks the manager whether he wants to add a new product, and after clicking the no button, a message is given that the product has been added to the stock.*



*Program management automatically updates the program ends after the product is added to the store and when the customer presses the Delete product from the store button on the interface, the product in the store is listed and when he clicks on the relevant product, the product is deleted and the manager receives a message that the product has been deleted.*



*Then, since the product was deleted from the store and I needed to show the price update process, I added it again and updated the product price by pressing the "Fiyat Güncelle" button on the interface.*



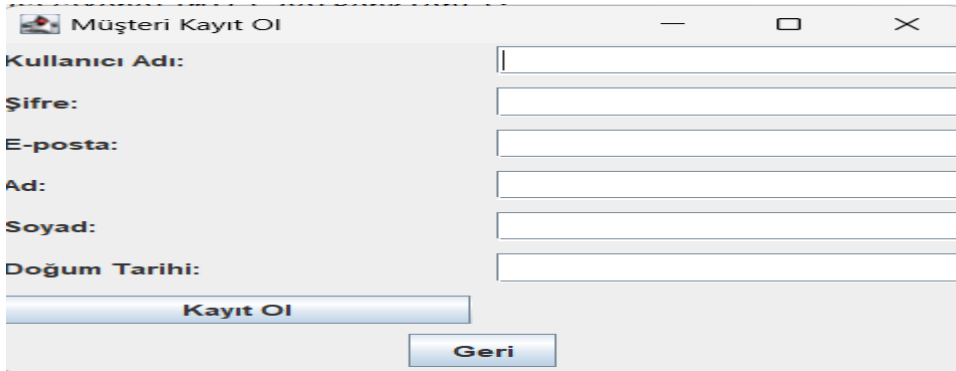
*Afterwards, I pressed the "Stock Control" button from the Admin interface and displayed the product with the new updated price information.*



The screenshot shows a window titled "Stok Kontrolü" with a standard Windows title bar (minimize, maximize, close buttons). The window content displays the following product information:

- Ürün Adı: Gömlek
- Beden: S
- Renk: Yeşil
- Stok: 12

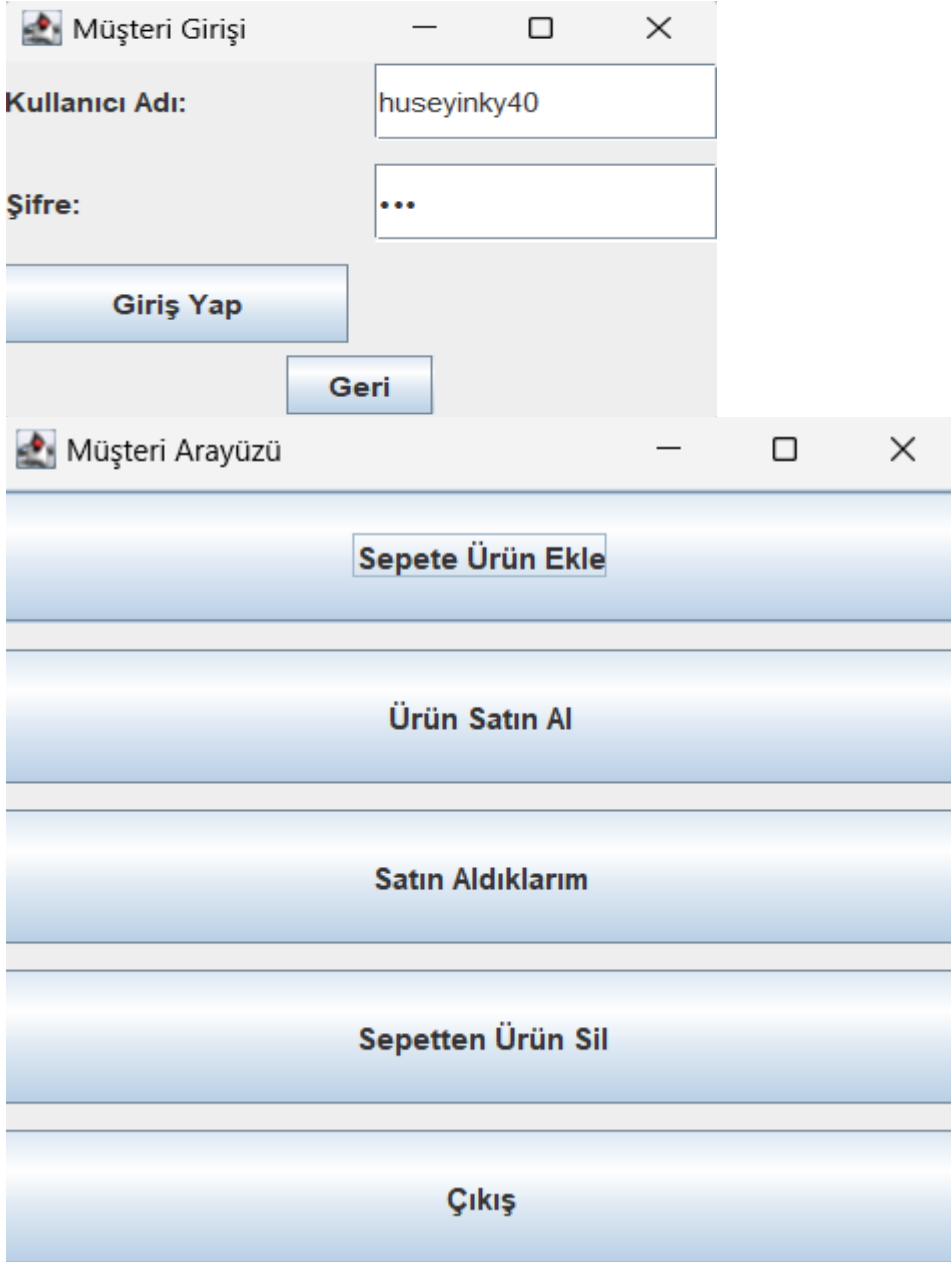
*If the "Kayıt ol" button is pressed on the Arel Giyim page, a new record will be created for the customer.*



The screenshot shows a window titled "Müşteri Kayıt Ol" with a standard Windows title bar. The window contains a registration form with the following fields and buttons:

- Kullanıcı Adı: (text input field)
- Şifre: (password input field)
- E-posta: (text input field)
- Ad: (text input field)
- Soyad: (text input field)
- Doğum Tarihi: (text input field)
- Kayıt Ol (button)
- Geri (button)

*Afterwards, the customer logs in with the registered username and password. Then the "Müşteri Arayüzü" screen opens.*



**Müşteri Girişi**

Kullanıcı Adı: huseyinky40

Şifre: ...

Giriş Yap

Geri

**Müşteri Arayüzü**

Sepete Ürün Ekle

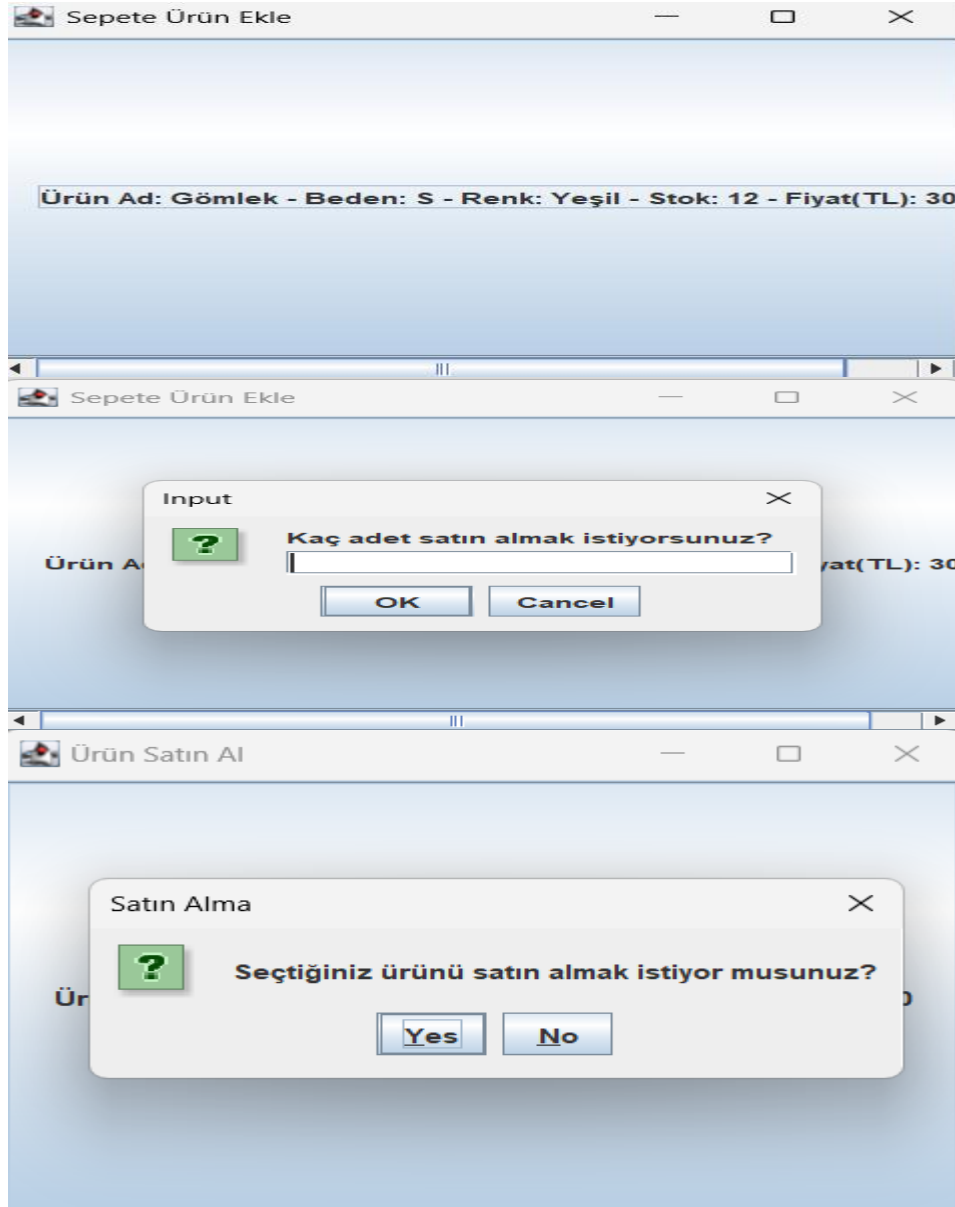
Ürün Satın Al

Satın Aldıklarım

Sepetten Ürün Sil

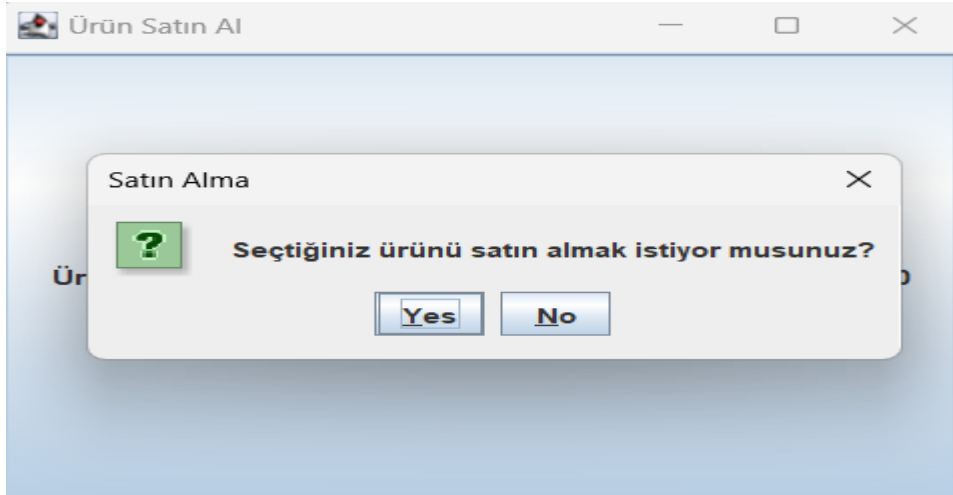
Çıkış

*Afterwards, when the customer presses the add product to cart button, the products that the manager has already added to the store are listed in front of the customer, and after he selects the product he wants, a window opens asking how many items he wants to add and the customer enters. Then, a message is given that the product has been successfully added to the cart.*





*Then, when you want to purchase the product in the basket, click on it.  
Then, after entering the payment and address information, the product is purchased successfully.*



The image shows a form titled 'Ödeme Ekranı' (Payment Screen) with a close button (X) in the top right corner. The form contains three input fields for payment information: 'Kart Numarası:' (Card Number) with the value '6833891205439016', 'CVV:' with the value '123', and 'Son Kullanma Tarihi:' (Expiration Date) with the value '12/24'. At the bottom of the form, there are two buttons: 'Ödeme Yap' (Make Payment) and 'İptal Et' (Cancel).

The image shows a form titled 'Adres Bilgileri' (Address Information) with a close button (X) in the top right corner. The form is divided into two sections: 'Müşteri Bilgileri' (Customer Information) and 'Adres Bilgileri' (Address Information). The 'Müşteri Bilgileri' section includes input fields for 'Ad:' (Name), 'Soyad:' (Surname), and 'Telefon:' (Phone). The 'Adres Bilgileri' section includes input fields for 'Adres Başlığı:' (Address Title), 'Şehir:' (City), 'İlçe:' (District), 'Mahalle:' (Neighborhood), 'Posta Kodu:' (Postal Code), and 'Adres:' (Address). At the bottom of the form, there is a button labeled 'Satin Al' (Purchase).