

# DoyDoy Food Ordering and Management System

## Design Document

---

Version: 1.0

### Document Revision History

Date	Version	ID	Author
25/05/2024	1.0	230303033	Fatih Çatalçam

### Contents

- 1. Introduction
  - 1.1. Purpose
  - 1.2. System Overview
  - 1.3. Design Objectives
  - 1.4. Definitions, Acronyms, and Abbreviations
- 2. Design Overview
  - 2.1. Introduction
  - 2.2. Environment Overview
  - 2.3. System Architecture
  - 2.4. Constraints and Assumptions
- 3. Interfaces and Data Stores
  - 3.1. System Interfaces
  - 3.2. Data Stores
- 4. Structural Design
  - 4.1. Design Explanation and Rationale

4.2. Class Diagram

4.3. Class Descriptions

5. Dynamic Model

5.1. Scenarios

## **1. Introduction**

### **1.1 Purpose**

This document provides a comprehensive design for the imaginary company DoyDoy Food Ordering and Management System, which is designed to manage the operations and control of the DoyDoy restaurant chain.

### **1.2 System Overview**

The DoyDoy Management System facilitates efficient management of restaurant operations including staff management, inventory control, and order processing.

### **1.3 Design Objectives**

- Usability: Provide an intuitive interface for restaurant managers and staff.
- Scalability: Ensure the system can scale with the growth of the restaurant chain.
- Security: Protect sensitive data such as employee information and financial records.

### **1.4 Definitions, Acronyms, and Abbreviations**

- OOP: Object-Oriented Programming
- GUI: Graphical User Interface

## **2. Design Overview**

### **2.1 Introduction**

This section provides an overview of the design considerations and the overall architecture of the DoyDoy Management System.

### **2.2 Environment Overview**

The system will operate in a client-server environment, where the client applications interact with a centralized server.

## 2.3 System Architecture

### 2.3.1 Top-level system structure

- Client Application: Interfaces for restaurant staff and management.
- Server Application: Handles business logic and data storage.

### 2.3.2 Subsystems

- Employee Management Subsystem
- Inventory Management Subsystem
- Order Processing Subsystem

## 2.4 Constraints and Assumptions

- The system will not include online ordering functionality.
- User authentication will be managed through predefined roles and permissions.

## 3. Interfaces and Data Stores

### 3.1 System Interfaces

#### 3.1.1 User Interface

The system will have graphical user interfaces (GUIs) for different types of users, including managers and regular staff.

#### 3.1.2 API Interfaces

APIs will be provided for integrating with third-party systems, such as payment gateways.

### 3.2 Data Stores

All data will be stored in a relational database. Key data stores include:

- employees.txt: Stores employee records.
- inventory.txt: Stores inventory records.
- orders.txt: Stores order records.

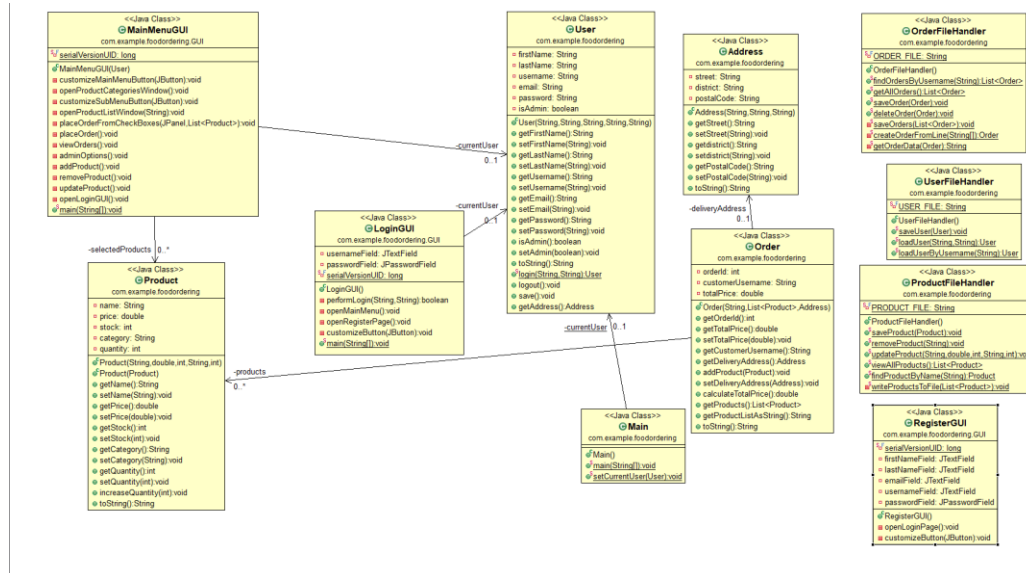
## 4. Structural Design

### 4.1 Design Explanation and Rationale

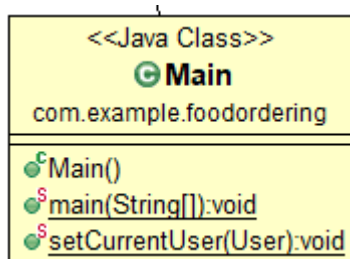
The DoyDoy Management System is designed using OOP principles, leveraging polymorphism and encapsulation to create a modular and maintainable codebase.

## 4.2 Class Diagram

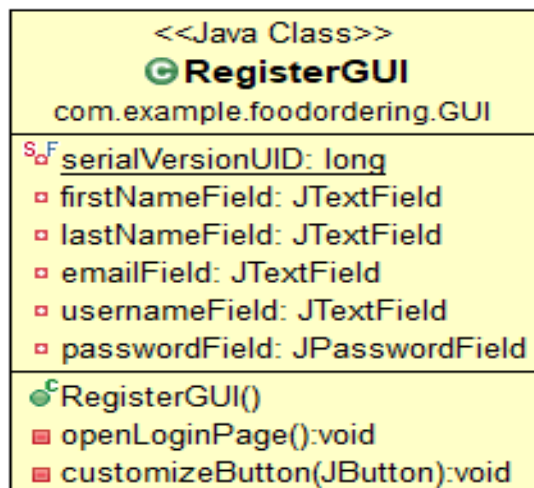
Class Diagram as a whole:















Main





















Register GUI













## Login GUI

<<Java Class>>	
 <b>LoginGUI</b>	
com.example.foodordering.GUI	
	usernameField: JTextField
	passwordField: JPasswordField
 	<u>serialVersionUID: long</u>
	<u>LoginGUI()</u>
	performLogin(String,String):boolean
	openMainMenu():void
	openRegisterPage():void
	customizeButton(JButton):void
 	<u>main(String[]):void</u>












## MainMenu GUI

<<Java Class>>	
 <b>MainMenuGUI</b>	
com.example.foodordering.GUI	
 	<u>serialVersionUID: long</u>
	<u>MainMenuGUI(User)</u>
	customizeMainMenuButton(JButton):void
	openProductCategoriesWindow():void
	customizeSubMenuButton(JButton):void
	openProductListWindow(String):void
	placeOrderFromCheckBoxes(JPanel,List<Product>):void
	placeOrder():void
	viewOrders():void
	adminOptions():void
	addProduct():void
	removeProduct():void
	updateProduct():void
	openLoginGUI():void
 	<u>main(String[]):void</u>















## UserFileHandler

<<Java Class>>	
 <b>UserFileHandler</b>	
com.example.foodordering	
 	<u>USER FILE: String</u>
	<u>UserFileHandler()</u>
 	<u>saveUser(User):void</u>
 	<u>loadUser(String,String):User</u>
 	<u>loadUserByUsername(String):User</u>




















## ProductFileHandler

<<Java Class>>	
 <b>ProductFileHandler</b>	
com.example.foodordering	
 	<u>PRODUCT FILE: String</u>
	<u>ProductFileHandler()</u>
	<u>saveProduct(Product):void</u>
	<u>removeProduct(String):void</u>
	<u>updateProduct(String,double,int,String,int):void</u>
	<u>viewAllProducts():List&lt;Product&gt;</u>
	<u>findProductByName(String):Product</u>
 	<u>writeProductsToFile(List&lt;Product&gt;):void</u>
















## OrderFileHandler

<<Java Class>>	
 <b>OrderFileHandler</b>	
com.example.foodordering	
 	<u>ORDER FILE: String</u>
	<u>OrderFileHandler()</u>
	<u>findOrdersByUsername(String):List&lt;Order&gt;</u>
	<u>getAllOrders():List&lt;Order&gt;</u>
	<u>saveOrder(Order):void</u>
	<u>deleteOrder(Order):void</u>
 	<u>saveOrders(List&lt;Order&gt;):void</u>
 	<u>createOrderFromLine(String[]):Order</u>
 	<u>getOrderData(Order):String</u>

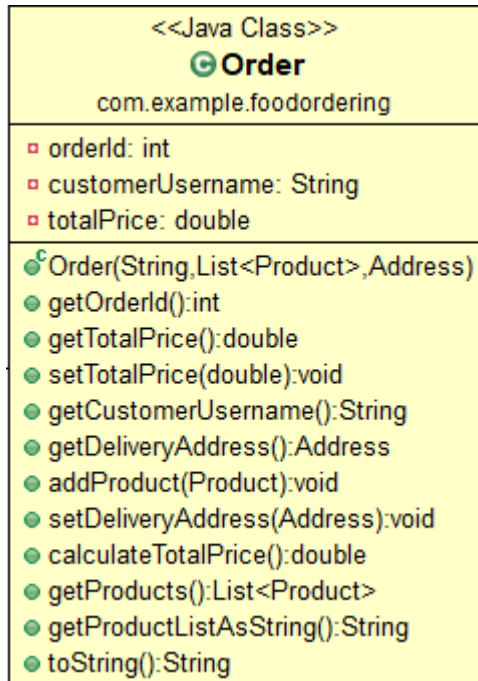
## User

<<Java Class>>	
 <b>User</b>	
com.example.foodordering	
<div><div>▣ firstName: String</div><div>▣ lastName: String</div><div>▣ username: String</div><div>▣ email: String</div><div>▣ password: String</div><div>▣ isAdmin: boolean</div></div>	
<div><div> User(String,String,String,String,String)</div><div> getFirstName():String</div><div> setFirstName(String):void</div><div> getLastName():String</div><div> setLastName(String):void</div><div> getUsername():String</div><div> setUsername(String):void</div><div> getEmail():String</div><div> setEmail(String):void</div><div> getPassword():String</div><div> setPassword(String):void</div><div> isAdmin():boolean</div><div> setAdmin(boolean):void</div><div> toString():String</div><div> <u>login(String,String):User</u></div><div> logout():void</div><div> save():void</div><div> getAddress():Address</div></div>	

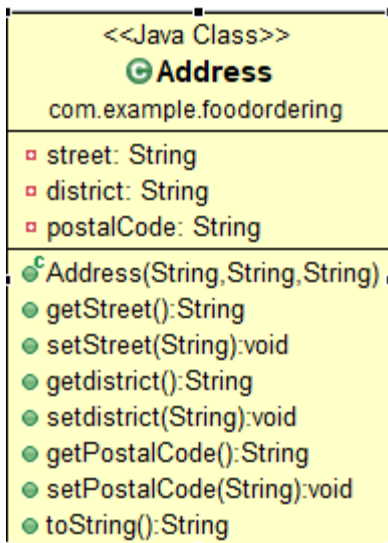
## Product

<<Java Class>>	
 <b>Product</b>	
com.example.foodordering	
<div><div>▣ name: String</div><div>▣ price: double</div><div>▣ stock: int</div><div>▣ category: String</div><div>▣ quantity: int</div></div>	
<div><div> Product(String,double,int,String,int)</div><div> Product(Product)</div><div> getName():String</div><div> setName(String):void</div><div> getPrice():double</div><div> setPrice(double):void</div><div> getStock():int</div><div> setStock(int):void</div><div> getCategory():String</div><div> setCategory(String):void</div><div> getQuantity():int</div><div> setQuantity(int):void</div><div> increaseQuantity(int):void</div><div> toString():String</div></div>	

## Order



## Address



## 4.3 Class Descriptions

### 4.3.1 Class: Employee

- Purpose: To model the relevant aspects of an employee.
- Attributes:
  - name (String): Stores the employee's name.



- id (String): Stores the unique employee ID.
- position (String): Stores the employee's position.
- Methods:
  - createEmployeeID(): Generates a unique employee ID.
  - getName(): Returns the employee's name.
  - setName(String name): Sets the employee's name.

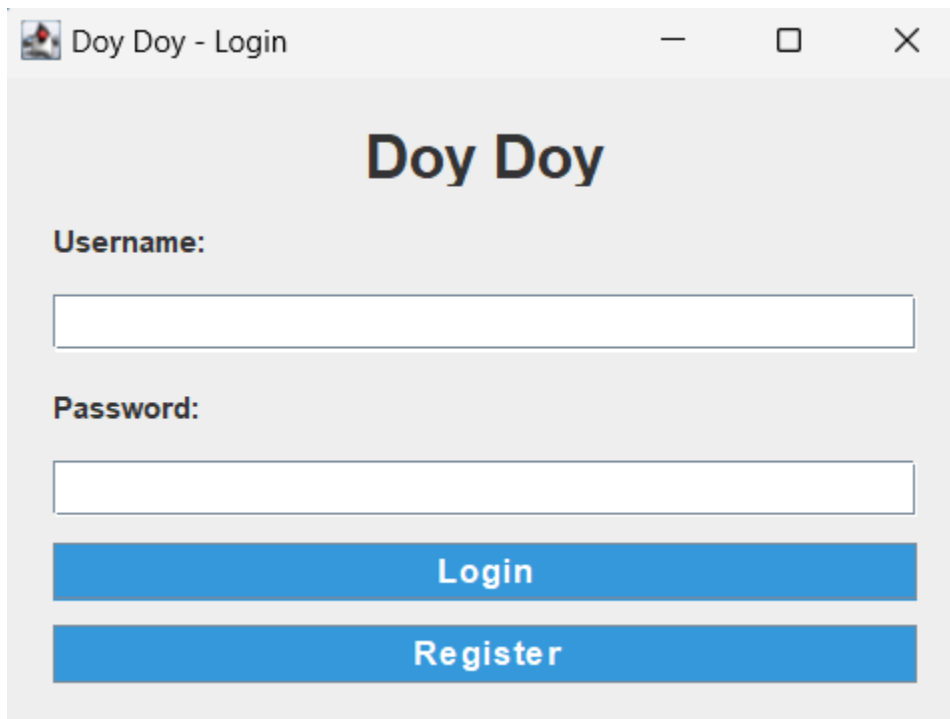
#### 4.3.2 Class: Manager (extends Employee)

- Purpose: To model the relevant aspects of a manager.
- Additional Attributes:
  - password (String): Stores the manager's password.
- Additional Methods:
  - createManagerPassword(): Generates a unique password for the manager.

## 5. Dynamic Model

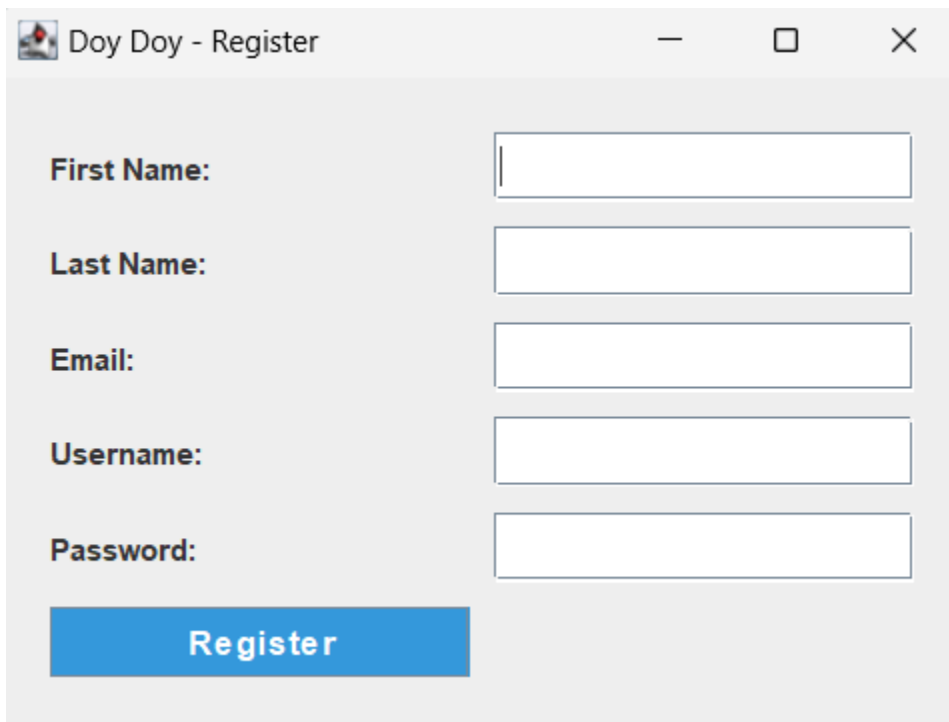
### 5.1 Scenarios

First off when we start the application The Login Page comes up.



The screenshot shows a window titled "Doy Doy - Login". The window has a light gray background. At the top, the text "Doy Doy" is displayed in a large, bold, black font. Below this, the label "Username:" is followed by a white text input field. Underneath the username field, the label "Password:" is followed by another white text input field. At the bottom of the window, there are two blue buttons with white text. The top button is labeled "Login" and the bottom button is labeled "Register". The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

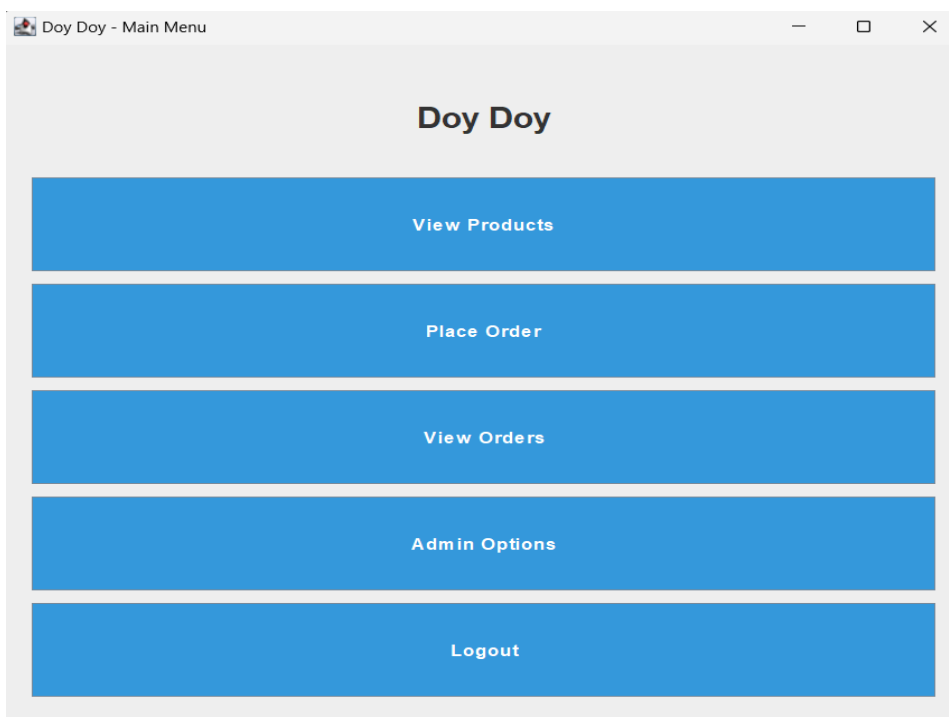
We need to register to be able to login the app, so we go to Register page.



A screenshot of a web application window titled "Doy Doy - Register". The window has a light gray background and standard window controls (minimize, maximize, close) in the top right corner. The registration form consists of five labeled input fields stacked vertically: "First Name:", "Last Name:", "Email:", "Username:", and "Password:". Each label is in bold black text, and each input field is a white rectangle with a thin gray border. Below the input fields is a blue rectangular button with the word "Register" in white text.

After we register, we go back to the login page again and login successfully.

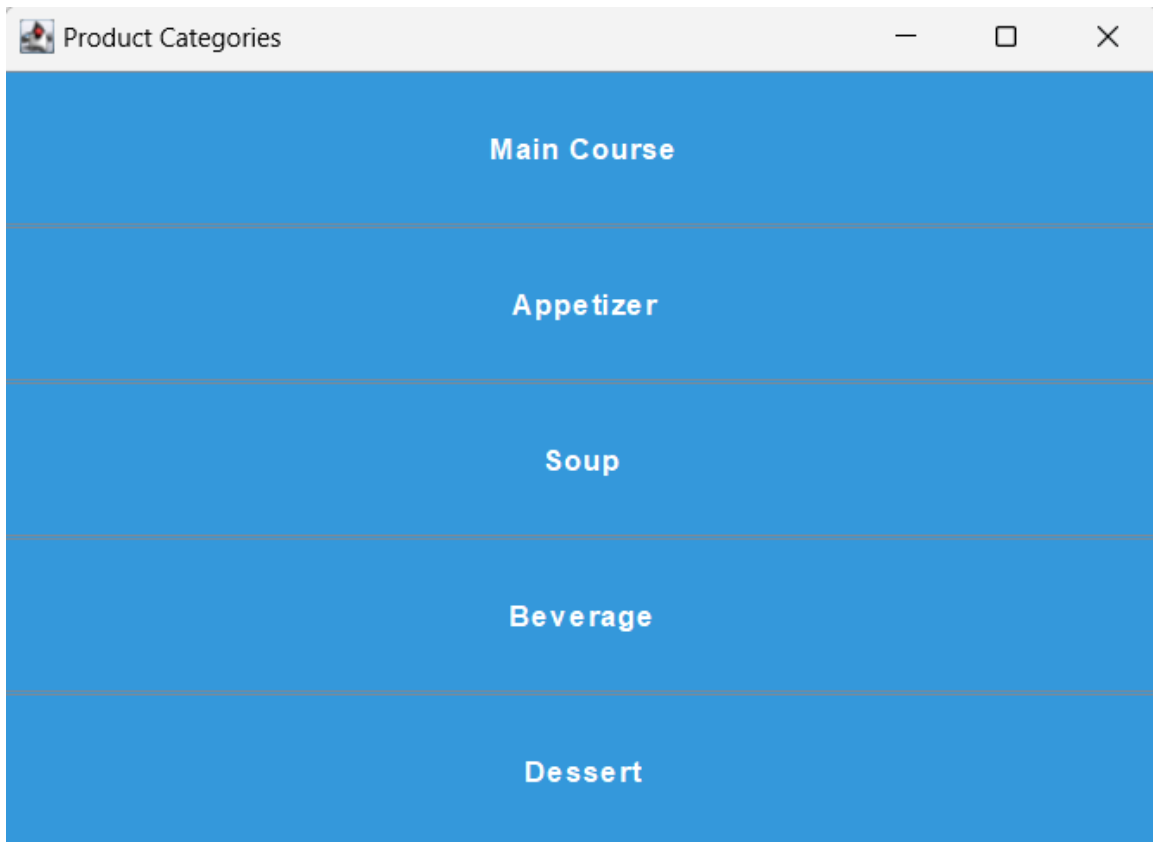
It takes us to the Main page.



A screenshot of a web application window titled "Doy Doy - Main Menu". The window has a light gray background and standard window controls in the top right corner. The main content area has a light gray background with the text "Doy Doy" centered at the top in bold black font. Below this, there are five blue rectangular buttons stacked vertically, each with white text: "View Products", "Place Order", "View Orders", "Admin Options", and "Logout".

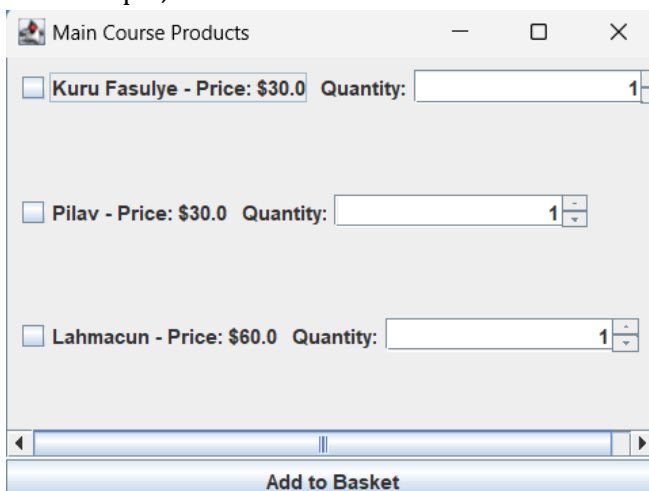
Here we can View Products that Administrator account added, Place orders, View Orders, Check out Admin Options and Logout.

Let's take a closer look to the View Products



We can see here our 5 main categories.

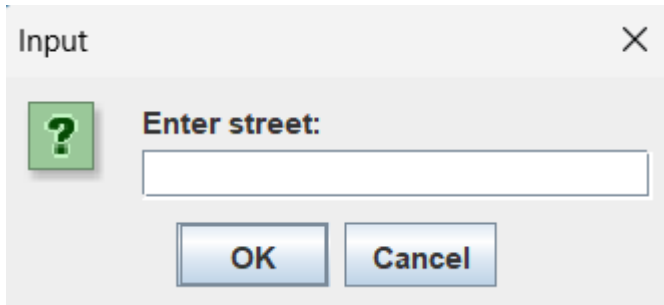
For example, let's take a look at Main Courses



Here we can choose the food and its quantity that we will be going to add to our basket.

Now let's order the food that we added to our basket.

First off it asks for us or address details.

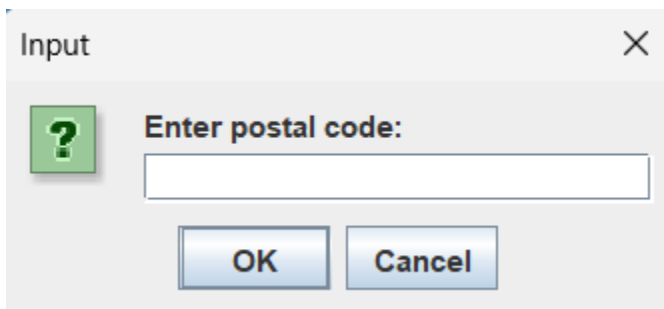


Input

?

Enter street:

OK Cancel



Input

?

Enter postal code:

OK Cancel



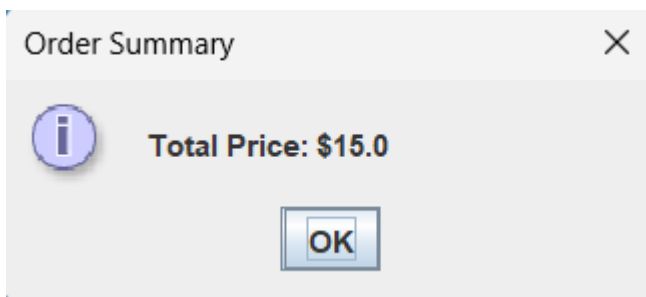
Input

?

Enter district:

OK Cancel

And then gives us the information of the price



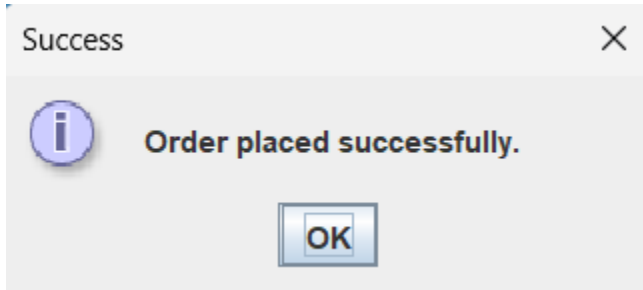
Order Summary

i

Total Price: \$15.0

OK

After we click OK, we get an information pop up.

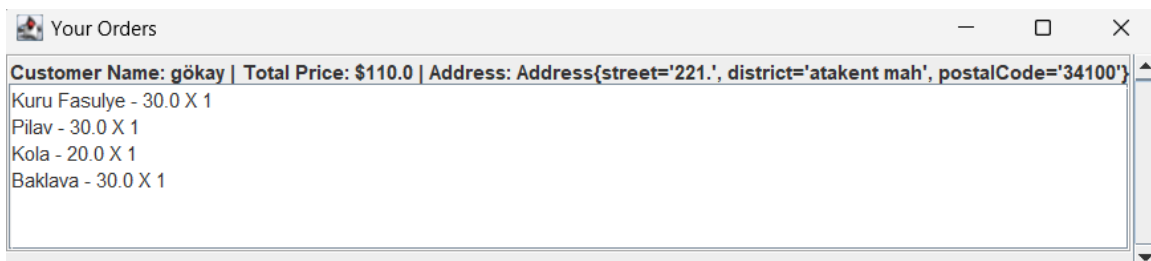


Now Its Time to View Our Order

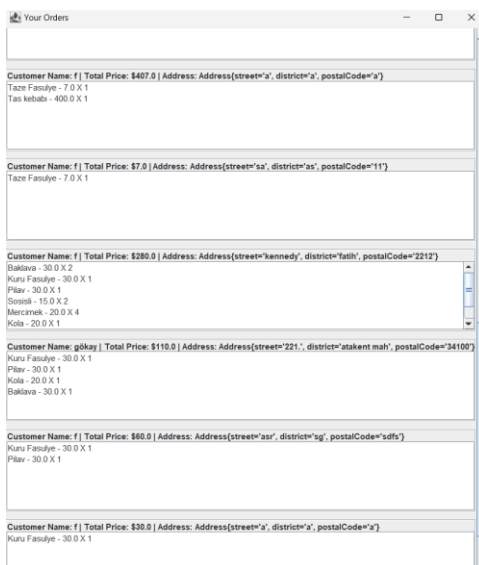
Normal users can only see their orders, but Administrator account can see all orders.

Here is the example.

This is what a normal account sees.

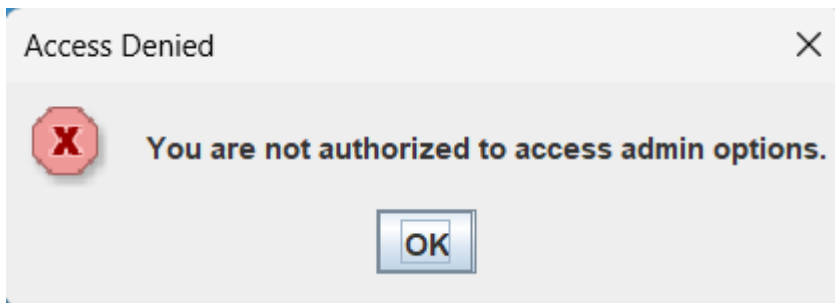


And this is what an administrator account sees.



Now let's get to the Admin Options

Normal users can't access admin options when they try to, they get an error like this.



Now this is what Administrator accounts see.

Admin accounts can Add products, Remove Products and update them.

