# **C:\Users\asus\AppData\Local\Microsoft\Windows\INetCache\Content.Word\muh-logo.png**DEUCENG-logo

**DOKUZ EYLÜL UNIVERSITY**

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CME 2210**

**Object Oriented Analysis and Design**

**ONLINE FOOD SERVICE SYSTEM**

**by**

**Melisa Beysümengü**

**İrem Çalmaz**

**İrem Okur**

**Contents**

[1](#_Toc39153534)

[**CHAPTER ONE** 3](#_Toc39153535)

[**INTRODUCTION** 3](#_Toc39153536)

[**1.1.** **Purpose** 3](#_Toc39153537)

[**1.2.** **Scope** 3](#_Toc39153538)

[**CHAPTER TWO** 4](#_Toc39153539)

[**REQUIREMENTS** 4](#_Toc39153540)

[**2.1. Object Oriented Model** 4](#_Toc39153541)

[**2.1.1 Address** 4](#_Toc39153542)

[**2.1.2 Phone** 4](#_Toc39153543)

[**2.1.3 Management** 4](#_Toc39153544)

[**2.1.4 User** 4](#_Toc39153545)

[**2.1.5 Customer** 5](#_Toc39153546)

[**2.1.6 Admin** 5](#_Toc39153547)

[**2.1.7 Restaurant** 5](#_Toc39153548)

[**2.1.8 Food** 6](#_Toc39153549)

[**2.2. Dependencies** 6](#_Toc39153550)

[**CHAPTER 3** 7](#_Toc39153551)

[**UML DIAGRAMS** 7](#_Toc39153552)

[**3.1 Class Diagram** 7](#_Toc39153553)

[**3.2 Use Case Diagram** 8](#_Toc39153554)

[**3.3 Activity Diagram I** 9](#_Toc39153555)

[**3.4 Activity Diagram II** 10](#_Toc39153556)

[**3.5 State Diagram I** 11](#_Toc39153557)

[**3.6 State Diagram II** 11](#_Toc39153558)

[**3.7 Sequence Diagram I** 12](#_Toc39153559)

[**3.8 Sequence Diagram II** 13](#_Toc39153560)

[**CHAPTER 4** 14](#_Toc39153561)

[**IMPLEMENTATION** 14](#_Toc39153562)

# **CHAPTER ONE**

# **INTRODUCTION**

## **Purpose**

Online food service system is created to help customers who wants to order food and the system keeps the name of the restaurants which has a package delivery option. In the past years, when a customer wanted to order a food, they would go to the restaurant or they would call the restaurant directly but this online food service system saves the customers’s time and assists them to choose the best food option according to their budget.

## **Scope**

The System is going to provide a safe, fast and easy online ordering system for customers. Every costumer has their own profiles and they can access to the system with given password. Costumers can choose a restaurant according to their budget and also they can see if a restaurant is full or not, if it is not full then they can see the number of customers in front of them. Payments will be done at the doorstep.

Furthermore, the system contains admin section. This section allows restaurant owners to add new food or remove an existing food on the current menu and admins can see the total orders. Moreover, admin can shut down the system to service because of the unexpected circumstances for instance occurness of over ordering.

# **CHAPTER TWO**

# **REQUIREMENTS**

## **2.1. Object Oriented Model**

The system has 8 classes. Such as Restaurant, Food, Customer, User, Address, Phone, Admin and Management.

### **2.1.1 Address**

Address class has 4 attributes. For instance streetname, town, city, description. These all are String. And this class includes their setters and getters.

### **2.1.2 Phone**

Phone class has 2 attributes. For instance country\_code, number. These all are String. And this class includes their setters and getters.

### **2.1.3 Management**

Management class created for managing the entire system. Functions will be added according to system’s needs.

### **2.1.4 User**

User class has 6 attributes. This class has: two String attribute called name, surname; an address attribute defined in Address class; an phone attribute defined in Phone class; a String attribute called password.

This class contains above attributes’s getters and setters ; toString() method which returns String.

### **2.1.5 Customer**

Customer class has 3 attributes. Such as two integer attribute called ID\_customer, count; an orders attribute as an ArrayList type.

This class has above attributes’s getters, setters and toString() method; DisplayAllOrders method which display all orders; CreateCustomer method which takes User class’s variables and add the new customer to the customerlist ArrayList;Order method takes 2 parameters from Restaurant and Food classes.

### **2.1.6 Admin**

Admin class has 2 attributes. Such as an integer attribute called ID\_admin, count .

This class has above attributes’s getters, setters and toString() method ; CreateAdmin method which takes User class’s variables and add the new admin to the adminlist ArrayList.

### **2.1.7 Restaurant**

Restaurant class has 7 attributes. This class has: two String attribute called restaurant\_name, ship\_min; an address attribute defined in Address class; an phone attribute defined in Phone class; and food attribute defined in Food class which type as a ArrayList; a customerqueue attribute using Queue structure.

This class contains above attributes’s getters and setters, toString(); Displayqueue method which shows all customers that ordered food from certain restaurant; addFood method which takes a parameter from Food class and add them to food ArrayList; RemoveFood method which removes it from food ArrayList.

### **2.1.8 Food**

Food class has 4 attributes. This class has: an String attribute called food\_name; an restaurant\_list attribute defined in Restaurant class which type as a ArrayList and lastly String attribute called ingredients using ArrayList structure also integer attribute called price.

This class contains above attributes’s getters and setters, toString(); addRestaurant method takes a parameter from Restaurant class and add them to restaurant\_list ArrayList.

## **2.2. Dependencies**

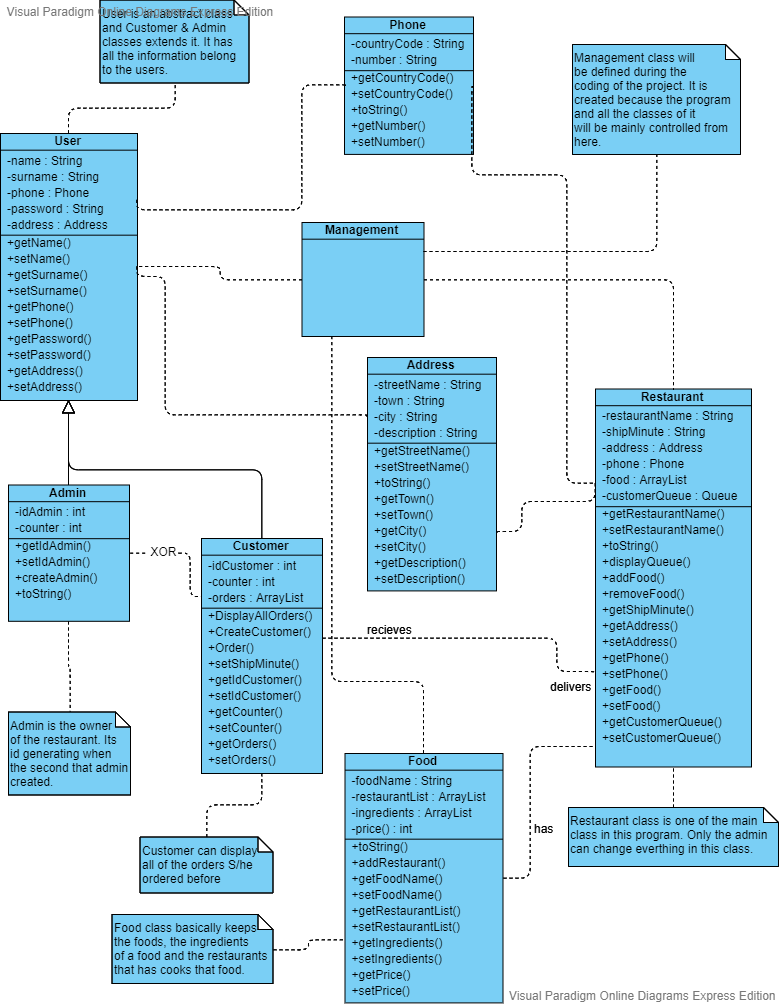
Admin and Customer Classes extends User Class. Also the User class implements Userinterface interface.

New functionalities can be added or remaining functionalities can be changed during the development process.

# **CHAPTER 3**

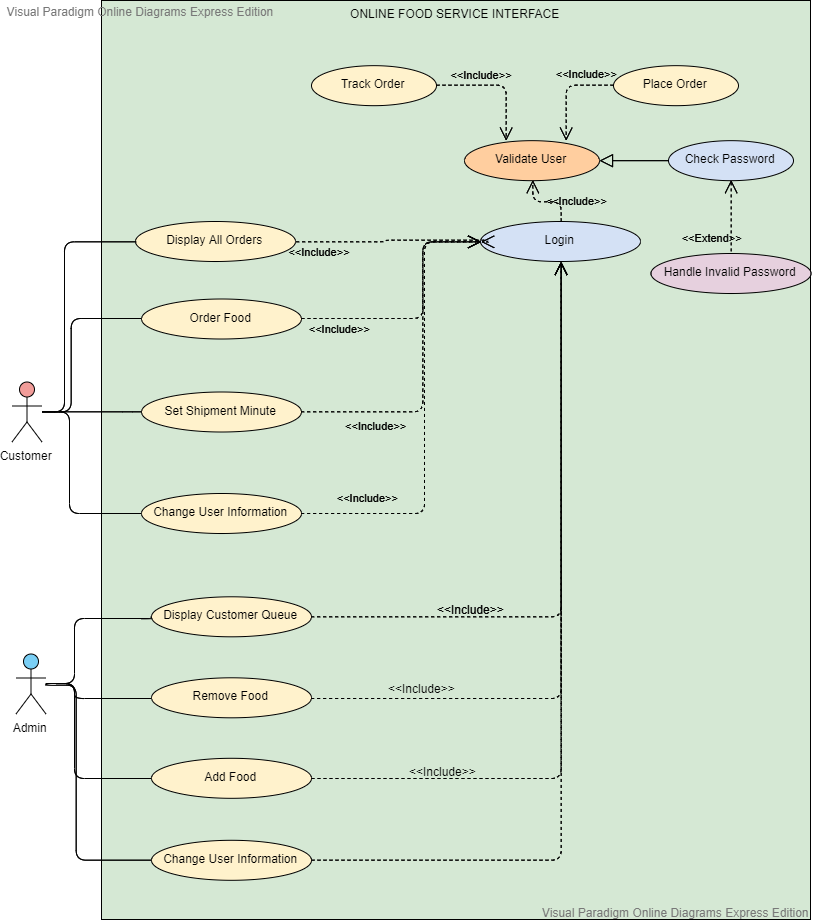
# **UML DIAGRAMS**

### **3.1 Class Diagram**

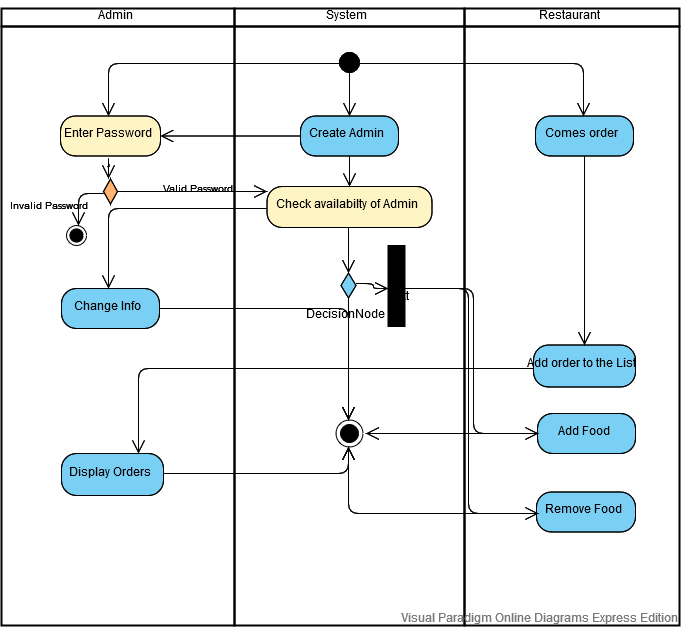


Class diagram shows the Online Food System's classes and the relation between them. As shown on the diagram, all of the classes interact with each other and some of them extends other classes. The system starts with the Management class and in this class the system’s aim is control all of the classes from here. Management class is empty right now because it will be implemented during the coding.

### **3.2 Use Case Diagram**

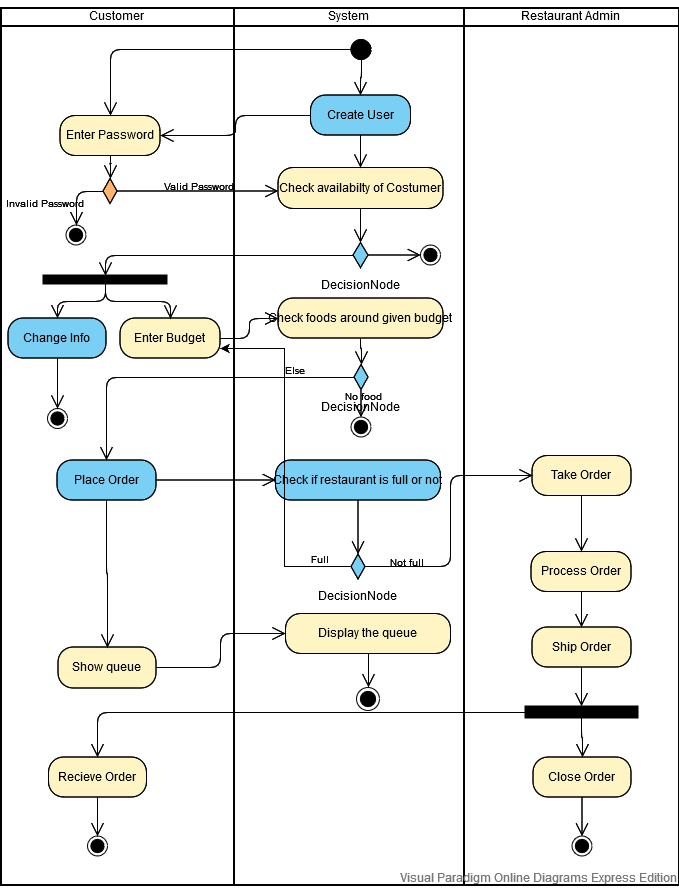


Use case diagrams represent the requirements of the system. They show how the users expect to interface with and get a benefit from the system through use cases. In this diagram there are two types of user which is customer and admin. Admin can display the queue, can change their information and add or remove the certain food.Customer can place order, display the orders can change their information.

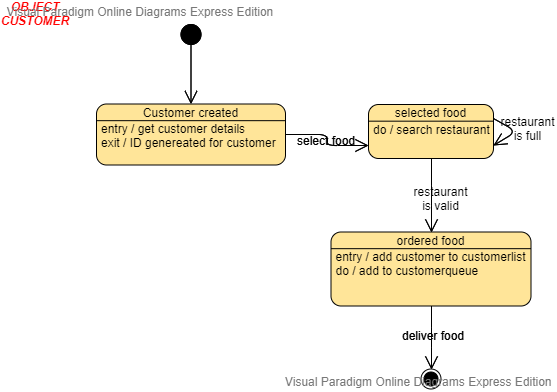
**3.3 Activity Diagram I**

Activity Diagram is basically a flowchart to represent the flow from one activity to another activity. In this diagram shows admin’s total of actions. Admin will enter password that is given by the system. Admin can also change their informations, can display all orders and add or remove the certain food.

### **3.4 Activity Diagram II**

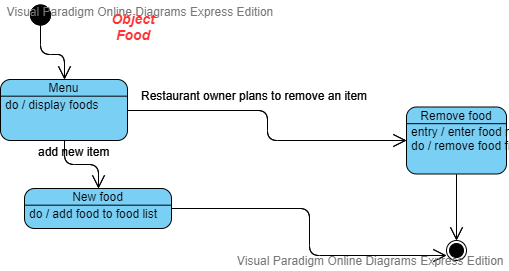


In this diagram shows customer’s total of actions. Customer will enter password that is given by the system. Customer can also change their informations. Restaurants will listed on screen considering their budget. Customer’s main action is place the order according to the restaurants availability.

**3.5 State Diagram I**

State Diagram describes all of the possible states that a particular object can get into. In this diagram lifetime of the customer object is shown. The lifetime of the object will start with creating customer , then continous with selecting food and ordering selected food.

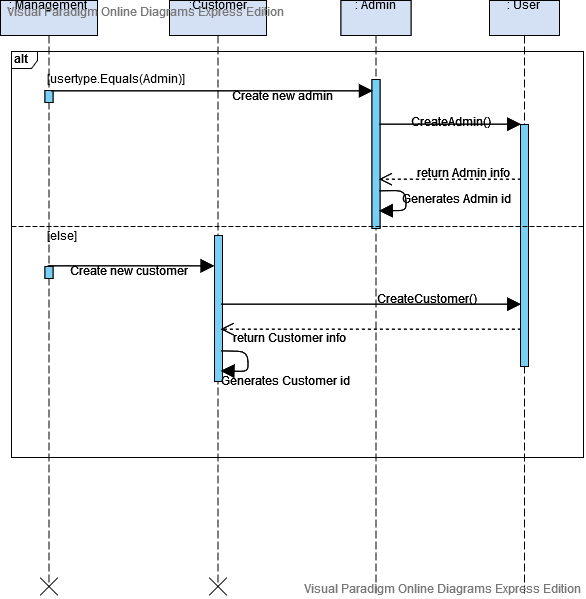
### **3.6 State Diagram II**



In this State Diagram, lifetime of the food object is shown. The lifetime of the object will start with adding the new food, or depending on the admin choice the program will remove or add the food.

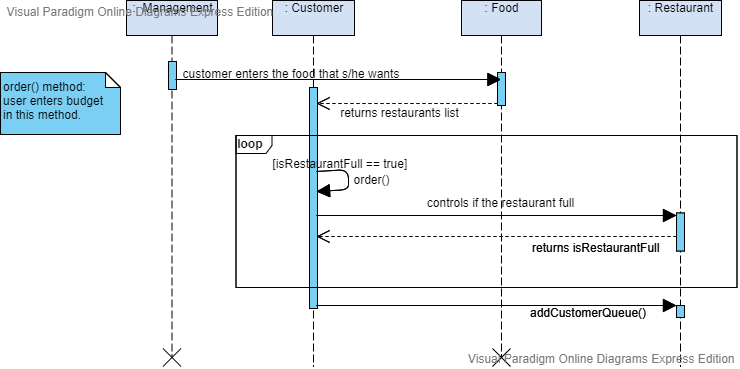
### **3.7 Sequence Diagram I**

***e***



Sequence Diagram is used to define event sequences between objects for a certain outcome.In management class, she/he should choose which type of user will be used in the system. If Customer is chosen, Customer Class will take informations from the User Class and then customer ID will be created randomly. However if Admin is chosen, Admin Class will take informations from the User Class and then admin ID will be created randomly.

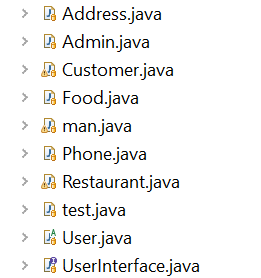
### **3.8 Sequence Diagram II**



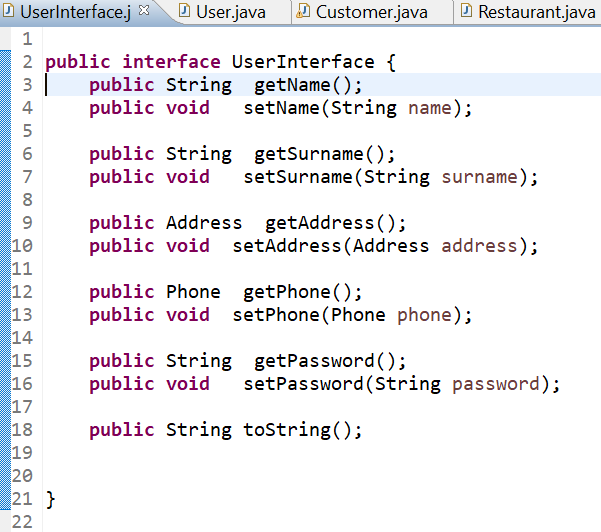
In this Sequence Diagram , customer’s ordering process is shown. Firstly, customer will enter name of food that they wants and food class will return the list of all restaurants that have the chosen food. Customer should choose the restaurant on that given list, moreover the system will check the restaurant availability. If the restaurant is available customer will be adding the end of the queue.

# **CHAPTER 4**

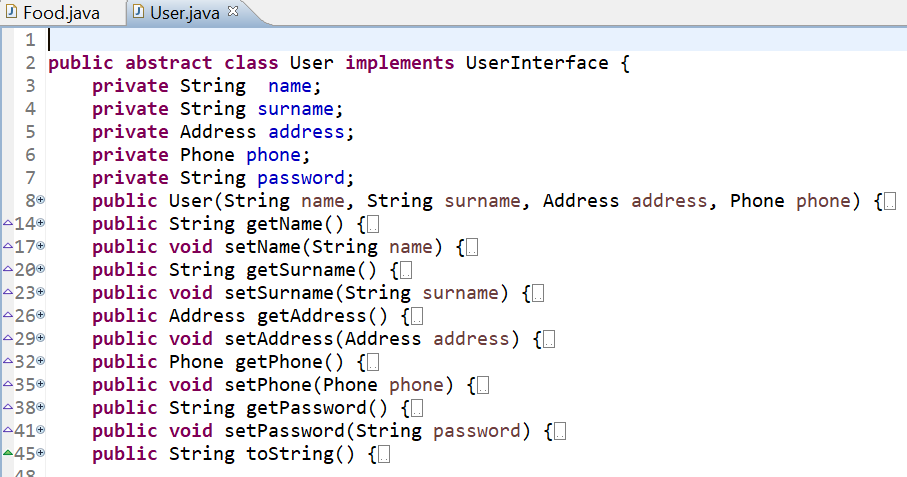
# **IMPLEMENTATION**

******

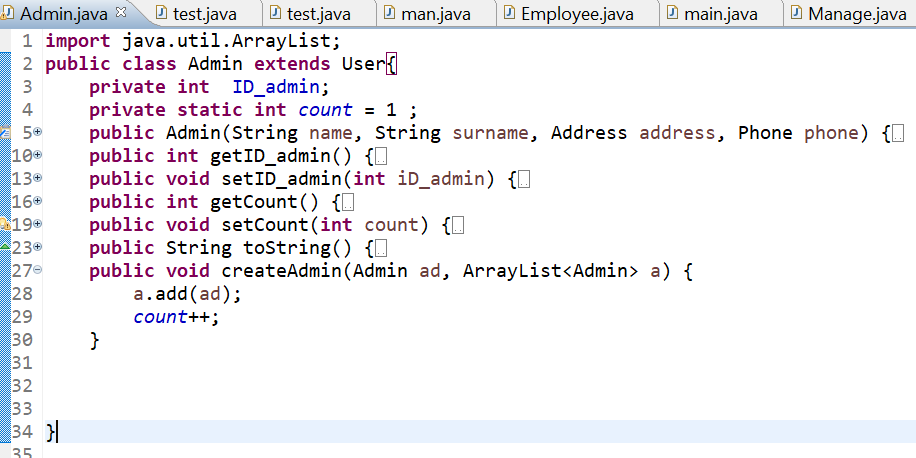
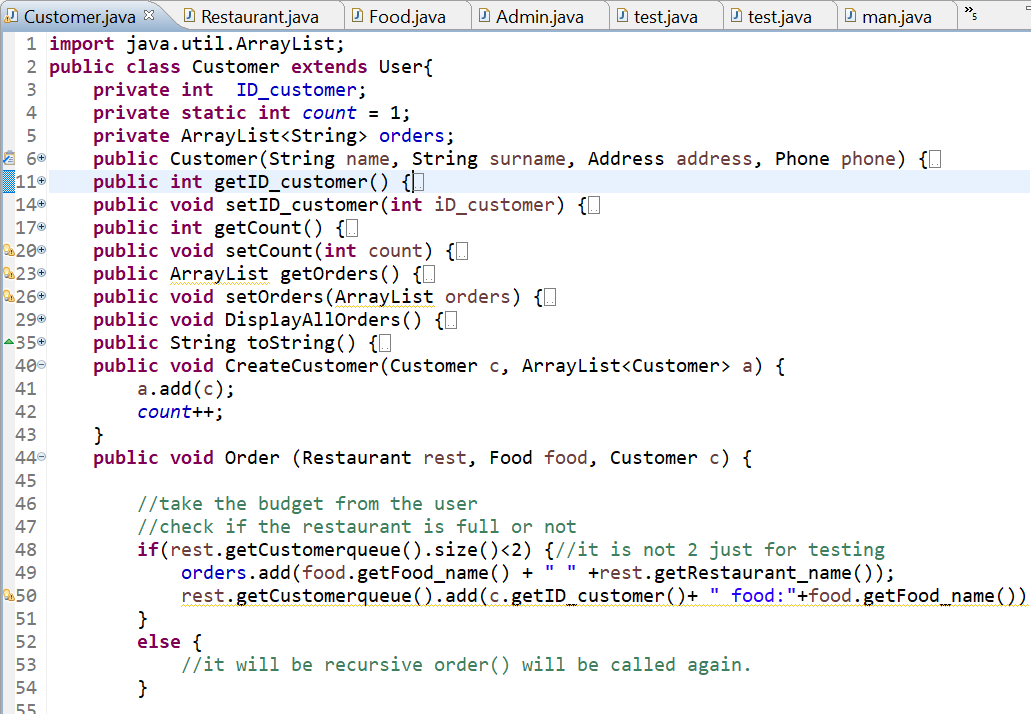
The system have 9 classes and 1 interface.

******

The UserInterface is a signature of the User Class. Signature means empty methods. The methods will implemented in the User Class.

******

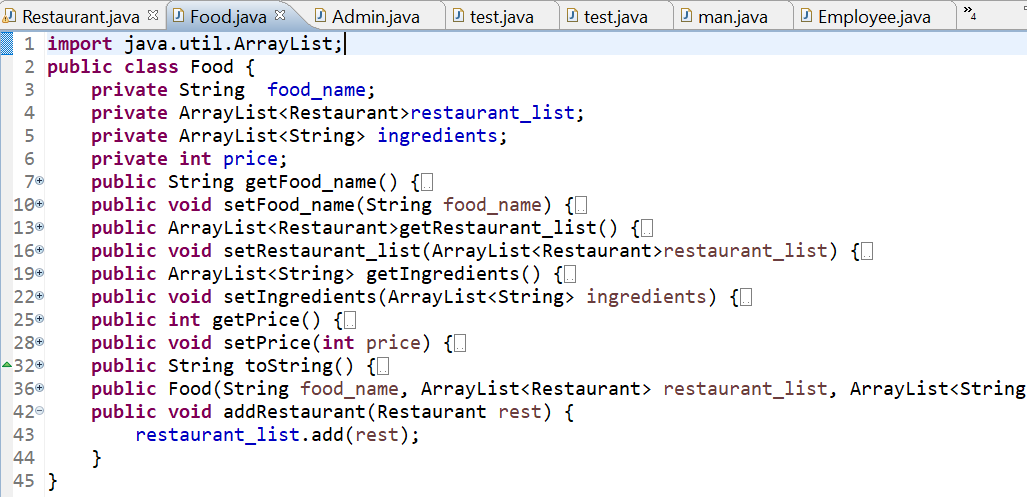
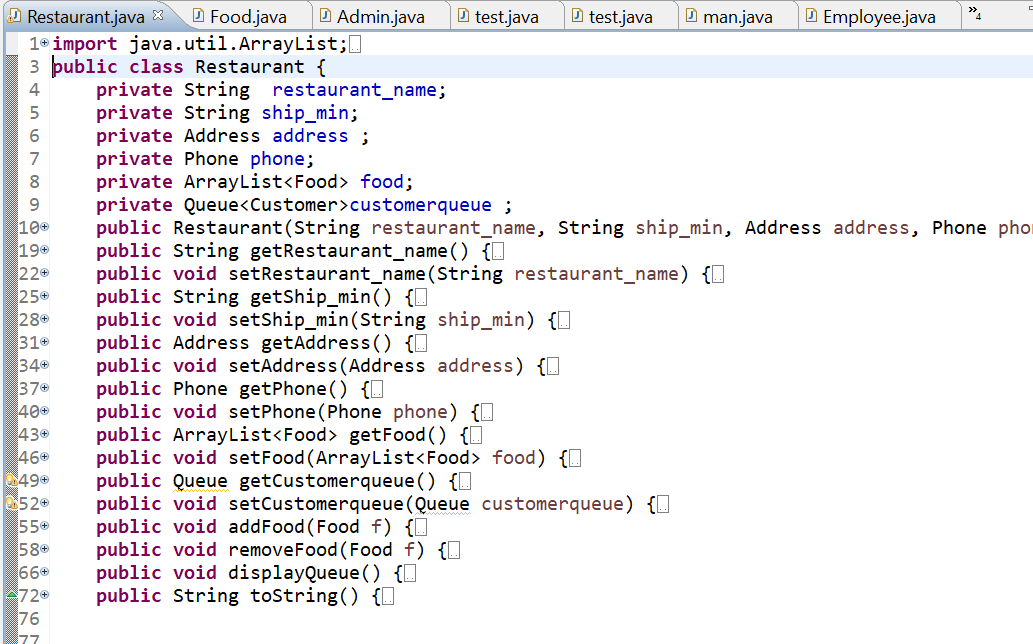
The User class implements UserInterface. Also User class’s attributes will be used in the Admin Class and the Customer Class.

******

Both Admin class and the Customer class extends User class. With the help of abstract class (superclass which is User class) the system has no extra attributes such as name,surname etc. for both class.

In the Customer class there are three methods which are DisplayQueue method , CreateCustomer method and last order method. Methods are explained in the chapter 2.

In the Admin class there is one method CreateAdmin method. This method also explained in the chapter 2.



In the Restaurant class there are three methods which are DisplayQueue method , addFood method and last removeFood method. Methods are explained in the chapter 2. The system keeps customers in Queue data structure.

In the Food class there is one method addRestaurant method. This method also explained in the chapter 2.