GTU Department of Computer Engineering CSE 222/505 - Spring 2022
GROUP 2
Final Report

Instructor
Prof. Dr. Fatih Erdoğan SEVİLGEN

PROJECT
Online Food Service System: HoldON

Table of Contents

Note: Updated parts from 2nd report are highligted with blue if an adding is performed, ~~scratch off~~ is performed if it is removed from project and explained why.

1. GROUP MEMBERS

STUDENT	ID
AZİZ CAN AKKAYA	1801042250
YASİN AKAR	215008003085
FEYZA NUR KÜÇÜK	1801042618
SERHAT SARI	200104004028
ABDULSAMED ASLAN	200104004098
MUHAMMED SİNAN PEHLİVANOĞLU	1901042664
ATACAN BAŞARAN	200104004008
ASUMAN SARE ERGÜT	1901042657

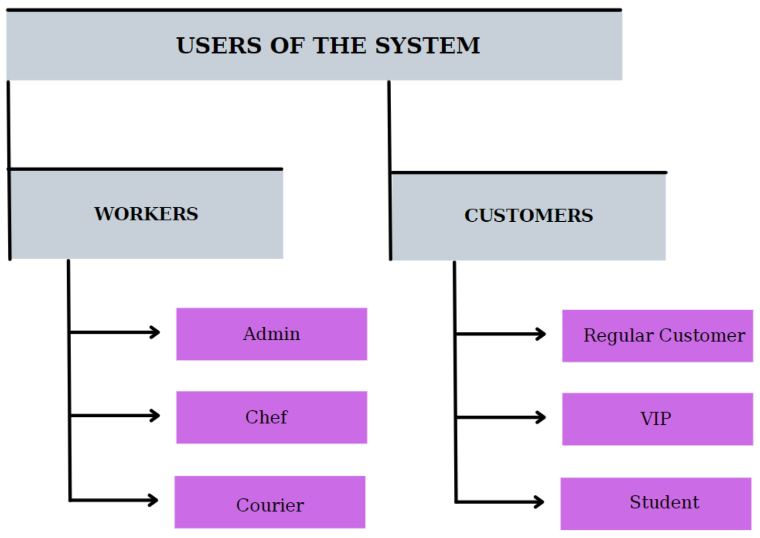
2. PROBLEM DEFINITION

Time is a determinant property nowadays that everyone considers while making decisions. In a short lunch break, since going to your favorite restaurant takes time, you may be obliged to prefer nearby restaurants. Or, in the group you will go to lunch with, there may be people who have different tastes. Giving different menus in groups and eating together still is possible in giving remote orders.

This time, let's have a look at from restaurant's perspective. Accommodate in a big place to serve more people is necessary. And bigger places costs higher bill and rent. This leads decreases in the profit of restaurant. To hold service quality maximum while given effort and time per customer is stable, is possible with online order system. Moreover, keeping customer data increases customer satisfaction due to it provides personalized menus.

Considering all those, an online food ordering system has been developed, named HoldON, referring “You don’t need to go anywhere to eat meal, just use your phone”.

3. USERS OF THE SYSTEM



3.a) Workers

3.a.1) Admin

Admin role is for maintaining and managing the whole restaurant system, it is designed to be owner of a restaurant. Hence, has capability of see and edit all information about restaurant itself, it's users and other features like orders, income/outcome.

He/she can see both type of user's - workers and customers- personal information, which are name, age, username and password and non-personal information of workers, such as certificate number. Salary of the worker is being determined by admin, according to customer's votes. In case of evaluated worker performance is decreased over some point, admin can fire that worker and can hiring someone new.

Managing the menu with Map of restaurant -create, change, delete whole menu or some foods inside menu - is also under the responsibility of admin.

Selecting the customer of month, according to records.(to implement SkipList)

3.a.2) Chef

Chef has same user information with additional professional qualification indicators like certificate number and experience year. It's responsibility is cooking the order in the waiting order queue. The status (beginner, junior, mid-level, senior) and salary of the Chef are updated according to professional competence, experience and customer votes.

3.a.3) Courier

Couriers deliver food orders that are prepared by Chef to customers and couriers can see the list of orders cooked to be delivered to customers. As a requirement of direct communication with the customer, unlike the Chef, Courirer has phone number information. The status (beginner, junior, mid-level, senior) and salary of the Courier are updated according to experience year and average score, rated by customers.

Delivering process will be prioritezed in the queue according to order's price, the most expensive order will be delivered first.

Also, distance between districts are calculated to minimize fuel consumption at the time of delivery.

3.b) Customers

For registration and verification process, a typical user is asked for name, age, job, unique username, password, phone number, balance information. Customer can see it's previous orders and can give new order from menu anytime. After order is taken, Customer gives votes to both Courirer and Chef. All customers can add alergetic property while login (AVL). And search for that material inside foods of the menu(MergeSort).

3.b.1) Regular Customer

Regular customer has no privilage like students or VIP customers. They are just typical users. Account is calculated without any reducer coefficent.

3.b.2) VIP

When the given order amount reaches the 5, regular customer turns into VIP customer and gains some privilages. Those are: Getting %15 discount for following orders and if order's price is high, gaining priority in the order queue.

3.b.3) Student

If the customer's job is selected as student during registration, without waiting to be VIP, customer's account will be evaluated with %25 discount.

4. REQUIREMENTS IN DETAILS

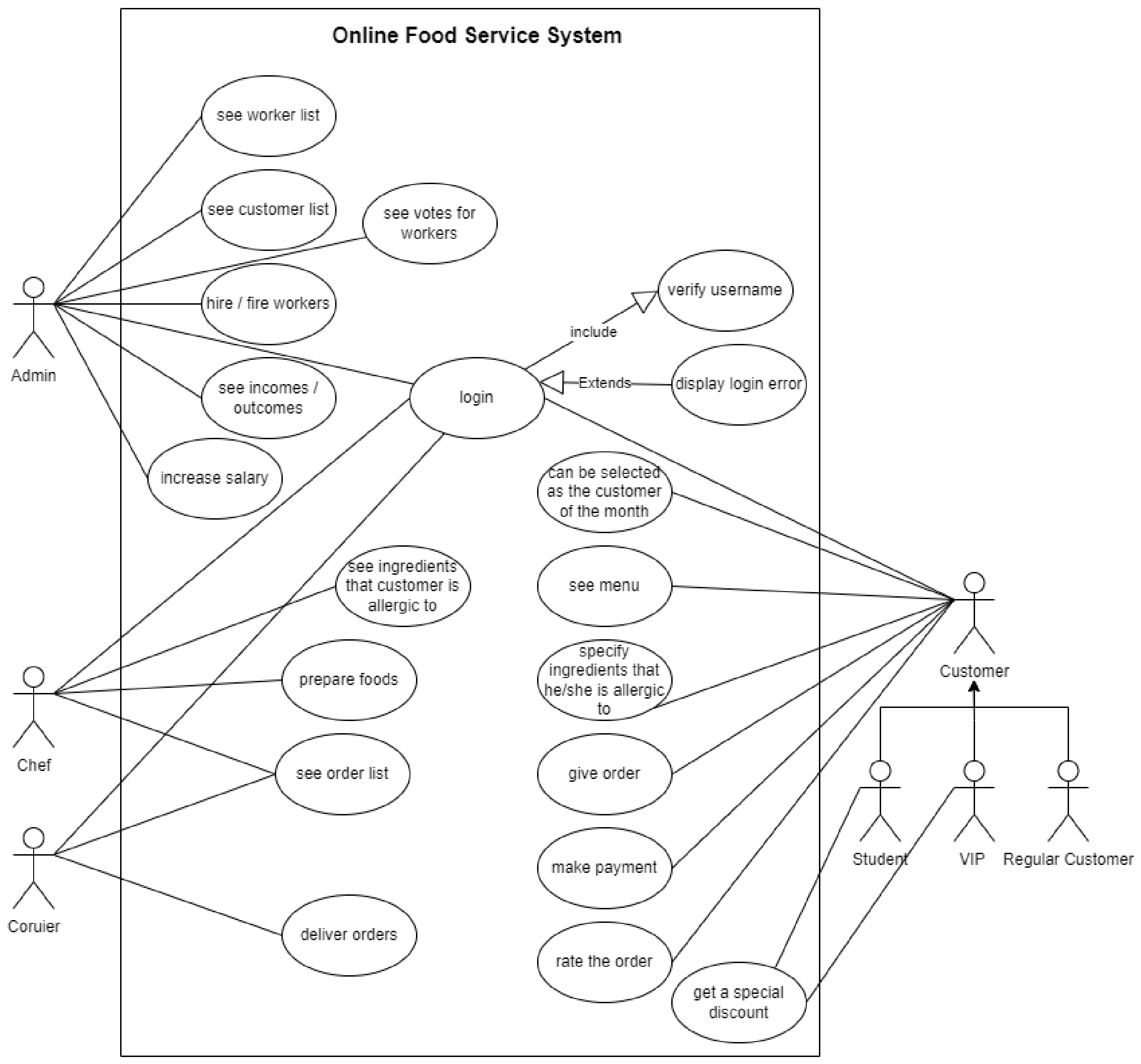
a. Functional Requirements

	ADMIN	COURIER	CHEF	STUDENT	REGULAR CUSTOMER	VIP
Login to the system	Y	Y	Y	Y	Y	Y
See the Menu	Y	N	N	Y	Y	Y
Edit the Menu	Y	N	N	N	N	N
Add/Remove Workers	Y	N	N	N	N	N
See worker information	Y	Y (only itself)	Y (only itself)	Y (not all)	Y (not all)	Y (not all)
Edit worker information	Y	Y (only itself)	Y (only itself)	N	N	N
Giving vote to workers	N	N	N	Y	Y	Y
See worker votes	Y	Y (only itself)	Y (only itself)	N	N	N
See order list	Y	Y	Y	N	N	N
Edit order list	Y	Y	N	N	N	N
Order a meal	N	N	N	Y	Y	Y
Gaining experience year	N	Y	Y	N	N	N
Gaining certificate	N	N	Y	N	N	N
Selecting customer of month	Y	N	N	N	N	N
Search for alergy inside foods	Y	N	N	Y	Y	Y
Being alergetic, add alergy	N	N	N	Y	Y	Y
Calculate district distances	N	Y	N	N	N	N
Discount in order	N	N	N	Y	N	Y

b. Non- Functional Requirements

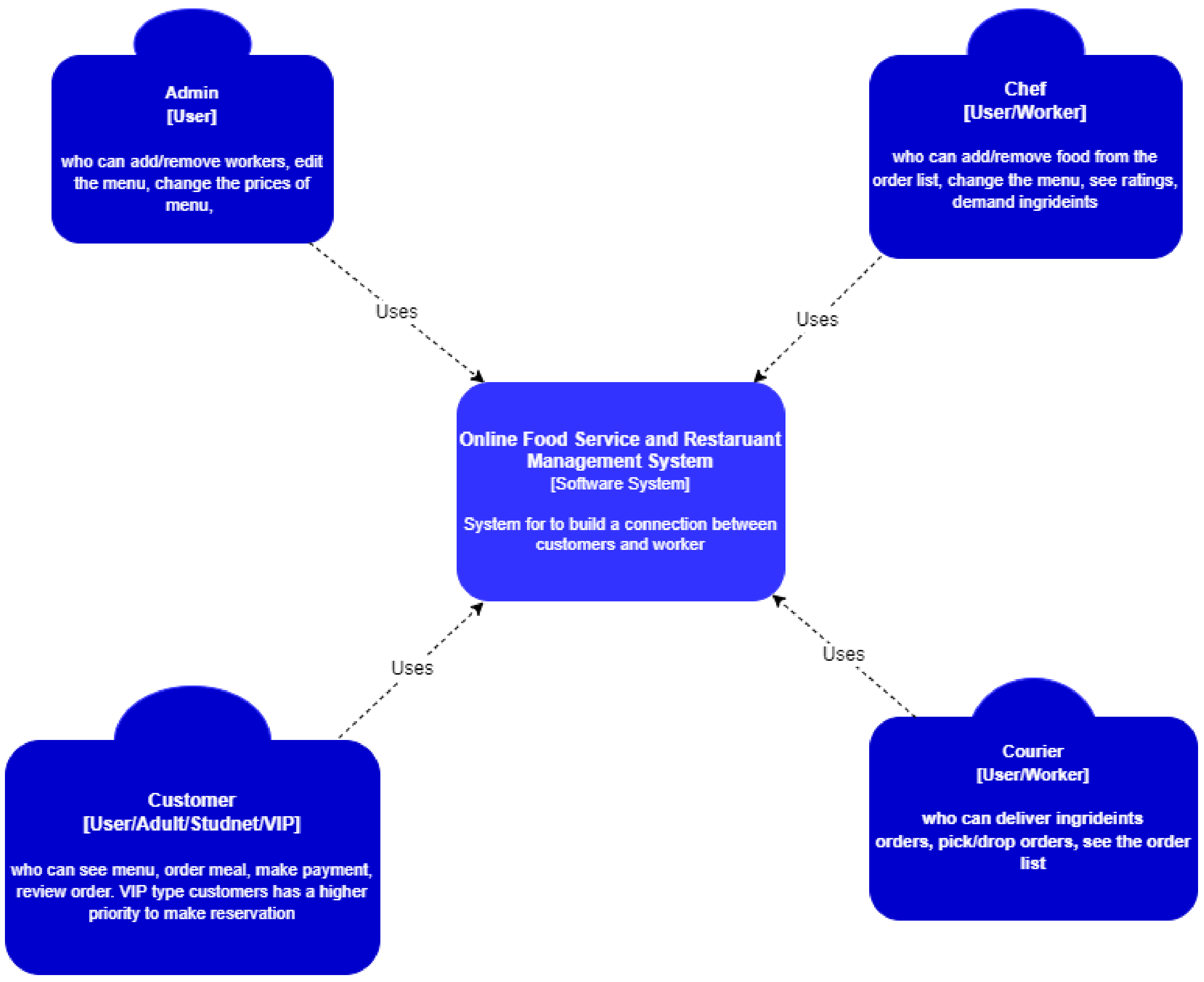
Back-end Software : Java 11
Software should be able to compile with “javac” on a linux distribution.
Hardware Interfaces: Mac, Linux and Windows operating systems.
The system follows a password policy.
The program should be maintainable.
The program should be usable. It should be safe and effective for the users to perform the desired tasks.
The program should be extendible. It should be able to receive new features and customizations for the upcoming versions.
System-Users’ information and restaurant information are kept in two separate files. When program is active, In case any login attempt by users, the system checks the validation of the user with the help of this user-information file. Other file should have restaurant information like menu and ingredients. In case any update on user information and restaurant information, related files is going to be updated simultaneously.

5. USE CASE DIAGRAM

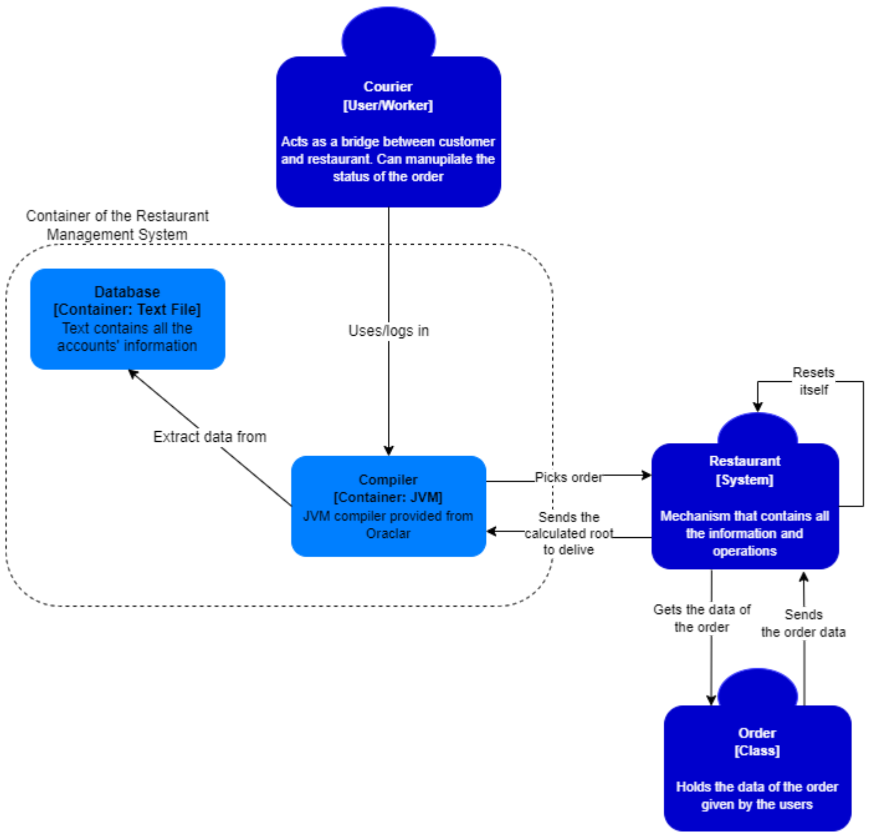


6. C4 MODEL OF THE SYSTEM

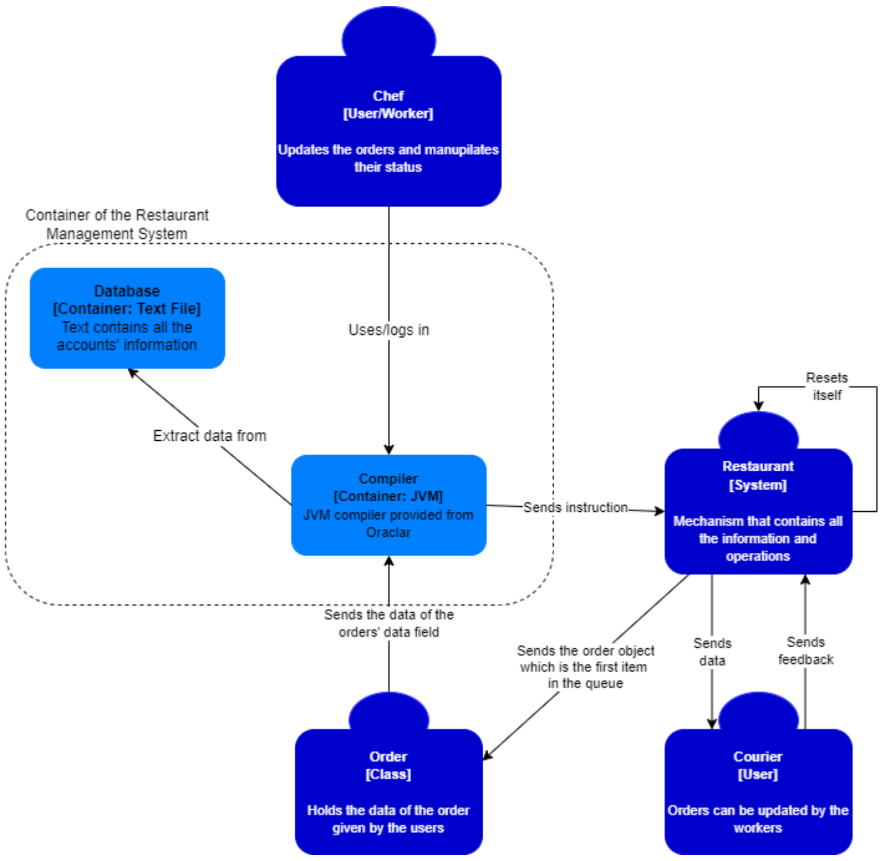
Level 1:


Admin
[User]
who can add/remove workers, edit the menu, change the prices of menu,
Chef
[User/Worker]
who can add/remove food from the order list, change the menu, see ratings, demand ingrideints
Uses
Uses
Online Food Service and Restaruant Management System
[Software System]
System for to build a connection between customers and worker
Uses
Uses
Customer
[User/Adult/Studnet/VIP]
who can see menu, order meal, make payment, review order. VIP type customers has a higher priority to make reservation
Courier
[User/Worker]
who can deliver ingrideints orders, pick/drop orders, see the order list

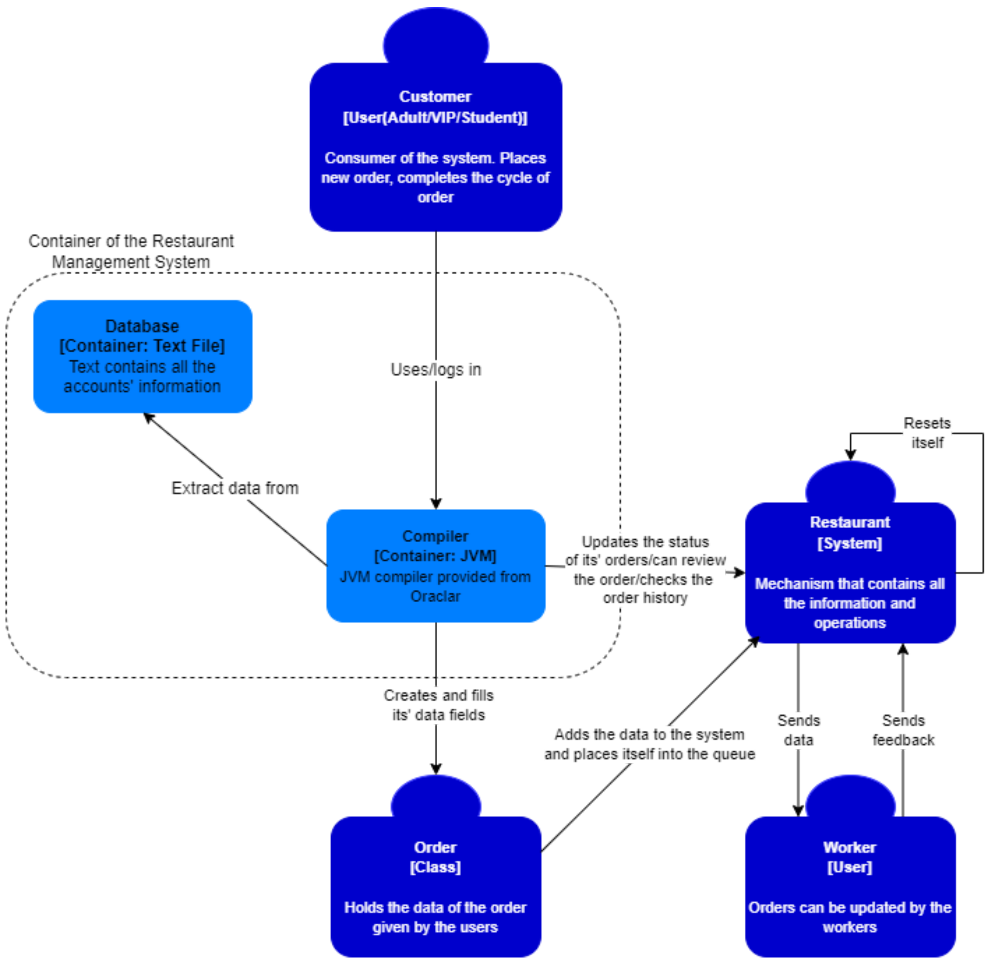
Level 2:



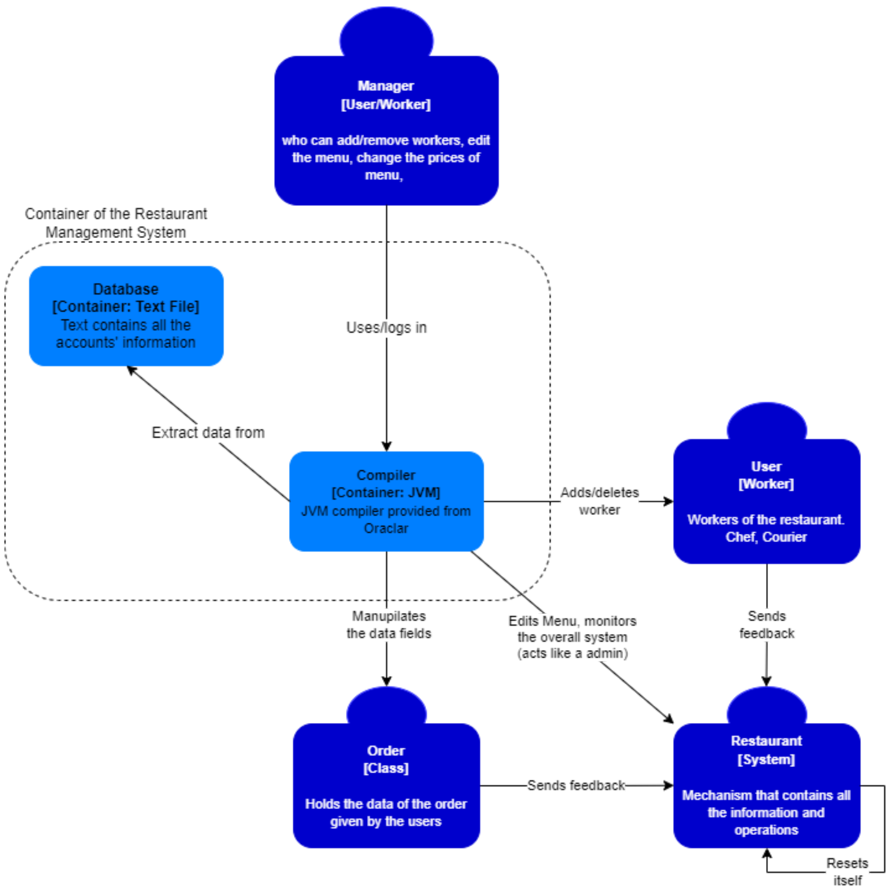
Courier
[User/Worker]
Acts as a bridge between customer and restaurant. Can manupilate the status of the order
Container of the Restaurant Management System
Database
[Container: Text File]
Text contains all the accounts' information
Uses/logs in
Extract data from
Compiler
[Container: JVM]
JVM compiler provided from Oraclar
Picks order
Sends the calculated root to delive
Resets itself
Restaurant
[System]
Mechanism that contains all the information and operations
Gets the data of the order
Sends the order data
Order
[Class]
Holds the data of the order given by the users



Chef
[User/Worker]
Updates the orders and manupilates their status
Container of the Restaurant Management System
Database
[Container: Text File]
Text contains all the accounts' information
Uses/logs in
Extract data from
Compiler
[Container: JVM]
JVM compiler provided from Oraclar
Sends instruction
Resets itself
Restaurant
[System]
Mechanism that contains all the information and operations
Sends the data of the orders' data field
Sends the order object which is the first item in the queue
Sends data
Sends feedback
Order
[Class]
Holds the data of the order given by the users
Courier
[User]
Orders can be updated by the workers

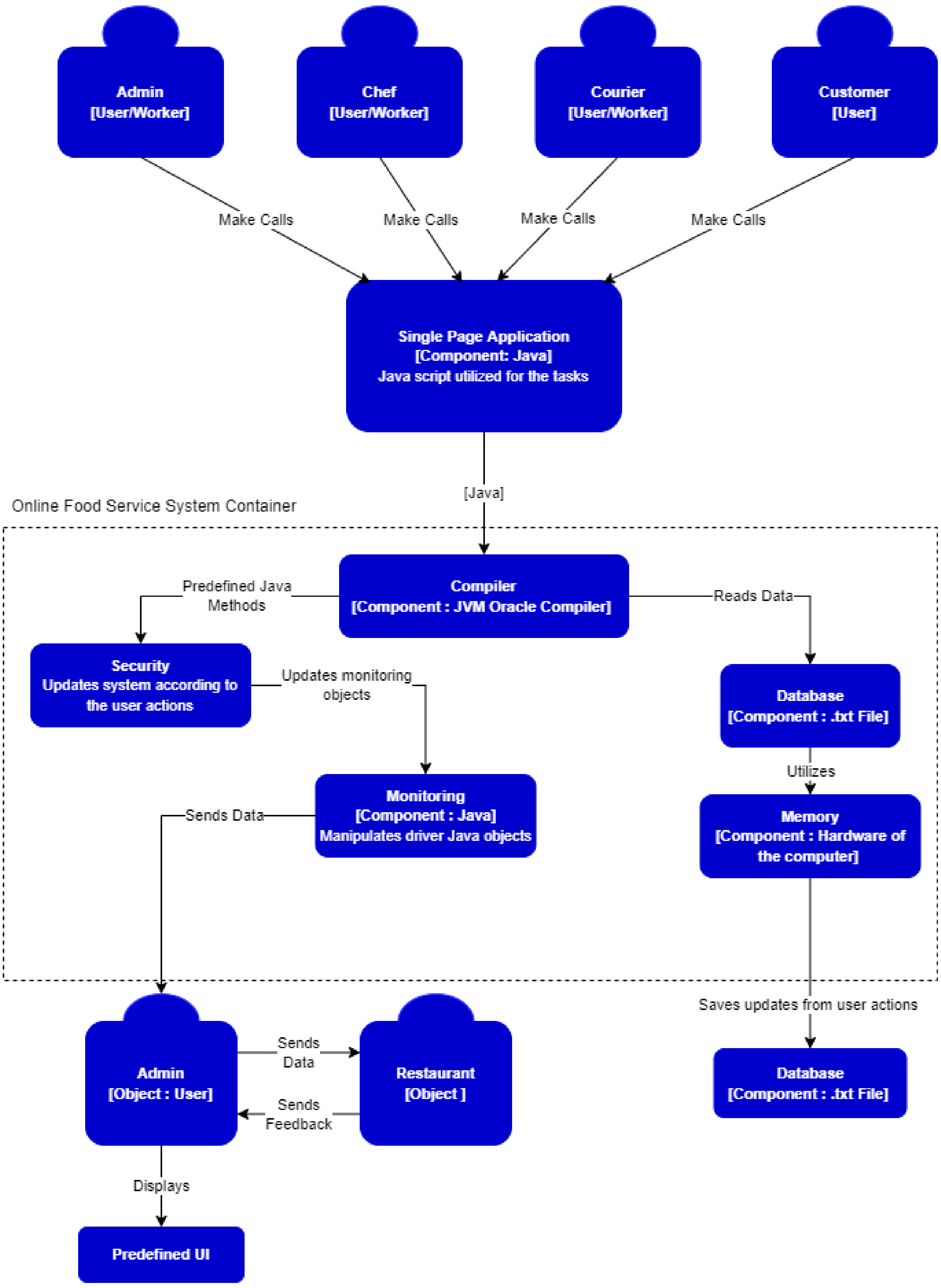


Customer
[User(Adult/VIP/Student)]
Consumer of the system. Places new order, completes the cycle of order
Container of the Restaurant Management System
Database
[Container: Text File]
Text contains all the accounts' information
Uses/logs in
Extract data from
Compiler
[Container: JVM]
JVM compiler provided from Oraclar
Updates the status of its' orders/can review the order/checks the order history
Resets itself
Restaurant
[System]
Mechanism that contains all the information and operations
Creates and fills its' data fields
Adds the data to the system and places itself into the queue
Sends data
Sends feedback
Order
[Class]
Holds the data of the order given by the users
Worker
[User]
Orders can be updated by the workers



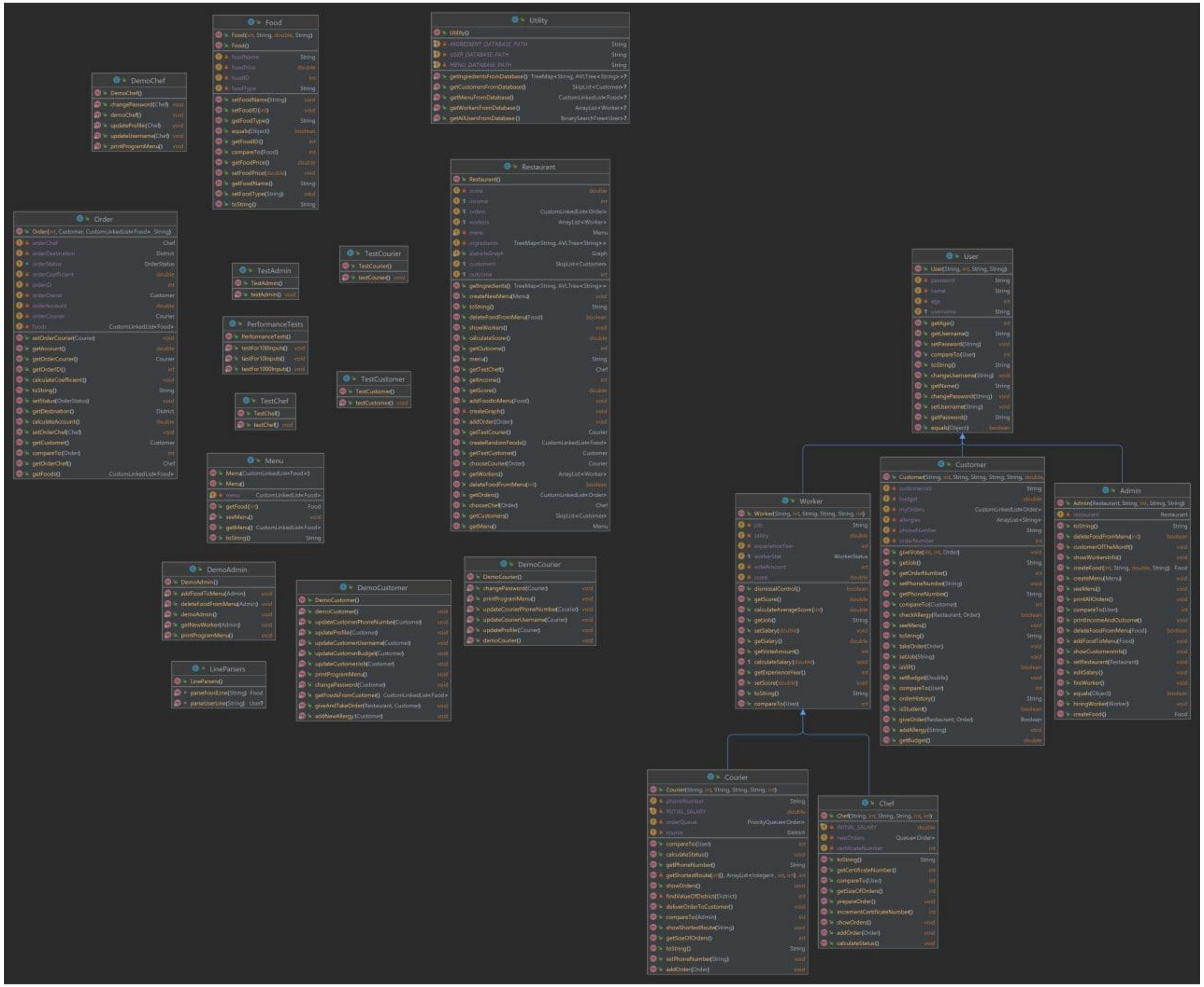
Manager
[User/Worker]
who can add/remove workers, edit the menu, change the prices of menu,
Container of the Restaurant Management System
Database
[Container: Text File]
Text contains all the accounts' information
Uses/logs in
Extract data from
Compiler
[Container: JVM]
JVM compiler provided from Oraclar
Adds/deletes worker
User
[Worker]
Workers of the restaurant. Chef, Courier
Manupilates the data fields
Edits Menu, monitors the overall system (acts like a admin)
Sends feedback
Order
[Class]
Holds the data of the order given by the users
Sends feedback
Restaurant
[System]
Mechanism that contains all the information and operations
Resets itself

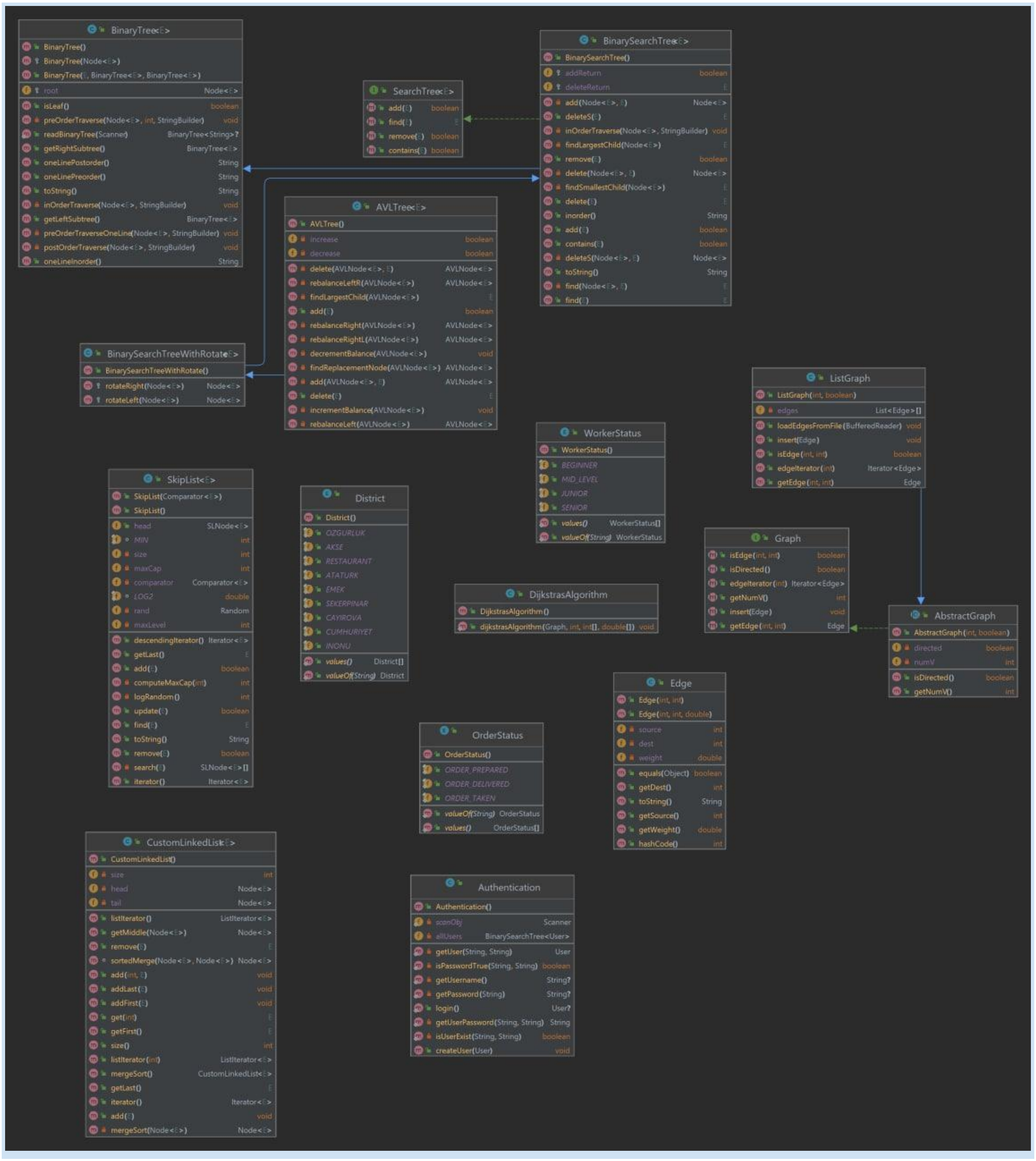
Level 3:



Admin [User/Worker]
Chef [User/Worker]
Courier [User/Worker]
Customer [User]
Make Calls
Make Calls
Make Calls
Make Calls
Single Page Application [Component: Java] Java script utilized for the tasks
[Java]
Online Food Service System Container
Compiler [Component : JVM Oracle Compiler]
Predefined Java Methods
Reads Data
Security Updates system according to the user actions
Updates monitoring objects
Database [Component : .txt File]
Utilizes
Monitoring [Component : Java] Manipulates driver Java objects
Sends Data
Memory [Component : Hardware of the computer]
Saves updates from user actions
Admin [Object : User]
Sends Data
Sends Feedback
Restaurant [Object]
Database [Component : .txt File]
Displays
Predefined UI

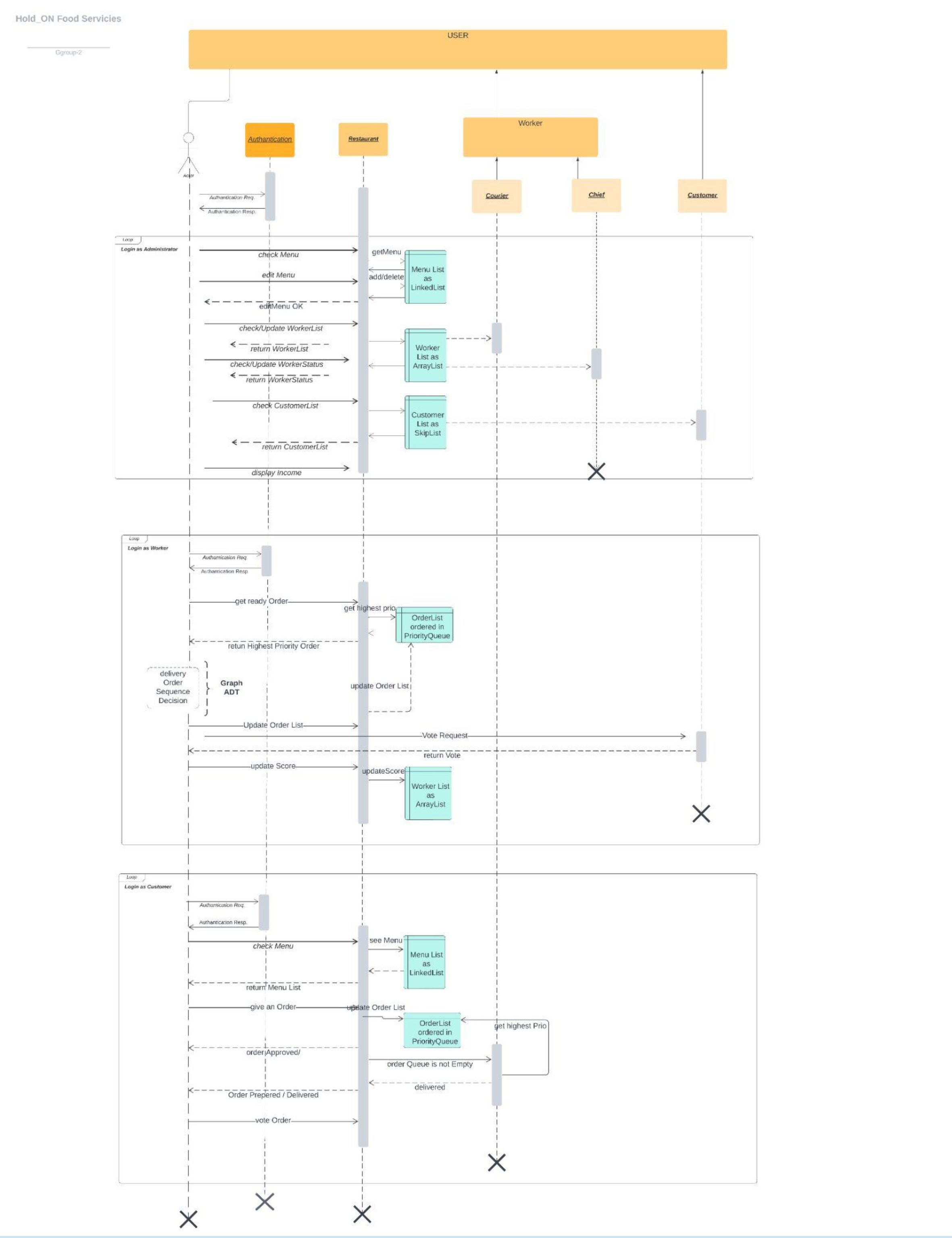
7. CLASS DIAGRAMS



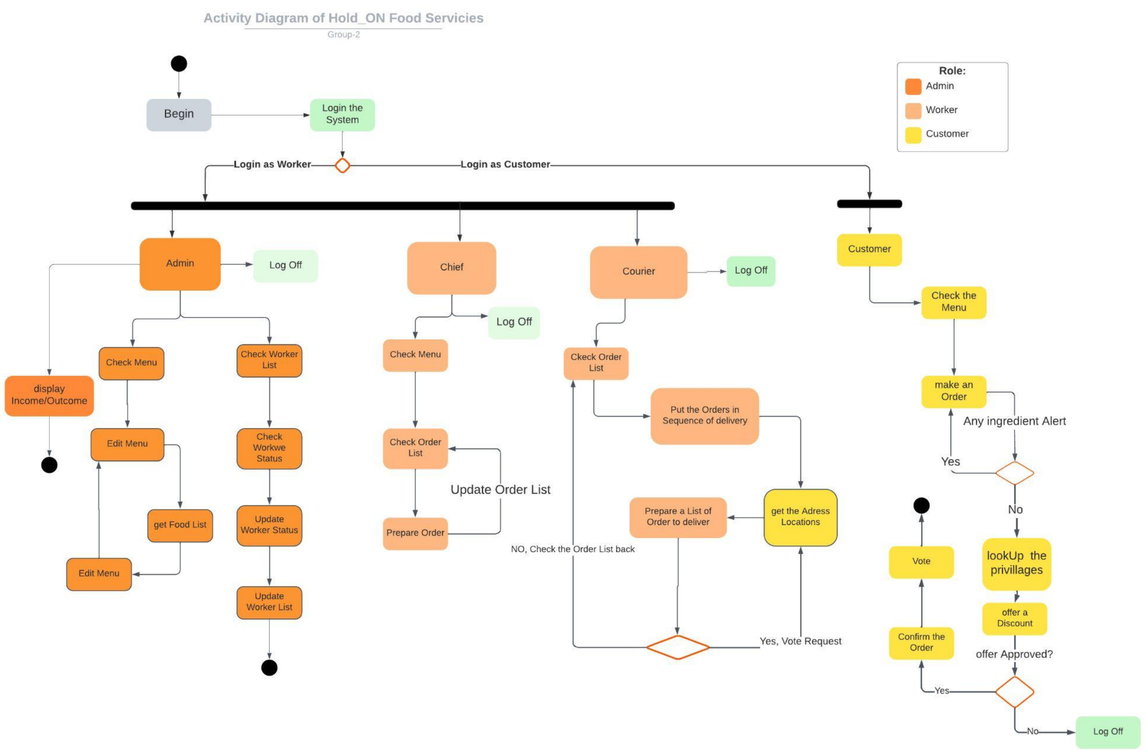


BinaryTree<E>
BinaryTree()
BinaryTree(Node<E>)
BinaryTree(E, BinaryTree<E>, BinaryTree<E>)
root Node<E>
isLeaf() boolean
preOrderTraverse(Node<E>, int, StringBuilder) void
readBinaryTree(Scanner) BinaryTree<String>?
getRightSubtree() BinaryTree<E>
oneLinePostorder() String
oneLinePreorder() String
toString() String
inOrderTraverse(Node<E>, StringBuilder) void
getLeftSubtree() BinaryTree<E>
preOrderTraverseOneLine(Node<E>, StringBuilder) void
postOrderTraverse(Node<E>, StringBuilder) void
oneLineInorder() String
SearchTree<E>
add(E) boolean
find(E) E
remove(E) boolean
contains(E) boolean
BinarySearchTree<E>
BinarySearchTree()
addReturn boolean
deleteReturn E
add(Node<E>, E) Node<E>
deleteS(E) E
inOrderTraverse(Node<E>, StringBuilder) void
findLargestChild(Node<E>) E
remove(E) boolean
delete(Node<E>, E) Node<E>
findSmallestChild(Node<E>) E
delete(E) E
inorder() String
add(E) boolean
contains(E) boolean
deleteS(Node<E>, E) Node<E>
toString() String
find(Node<E>, E) E
find(E) E
AVLTree<E>
AVLTree()
increase boolean
decrease boolean
delete(AVLNode<E>, E) AVLNode<E>
rebalanceLeftR(AVLNode<E>) AVLNode<E>
findLargestChild(AVLNode<E>) E
add(E) boolean
rebalanceRight(AVLNode<E>) AVLNode<E>
rebalanceRightL(AVLNode<E>) AVLNode<E>
decrementBalance(AVLNode<E>) void
findReplacementNode(AVLNode<E>) AVLNode<E>
add(AVLNode<E>, E) AVLNode<E>
delete(E) E
incrementBalance(AVLNode<E>) void
rebalanceLeft(AVLNode<E>) AVLNode<E>
BinarySearchTreeWithRotate<E>
BinarySearchTreeWithRotate()
rotateRight(Node<E>) Node<E>
rotateLeft(Node<E>) Node<E>
ListGraph
ListGraph(int, boolean)
edges List<Edge>[]
loadEdgesFromFile(BufferedReader) void
insert(Edge) void
isEdge(int, int) boolean
edgeIterator(int) Iterator<Edge>
getEdge(int, int) Edge
WorkerStatus
WorkerStatus()
BEGINNER
MID_LEVEL
JUNIOR
SENIOR
values() WorkerStatus[]
valueOf(String) WorkerStatus
SkipList<E>
SkipList(Comparator<E>)
SkipList()
head SLNode<E>
MIN int
size int
maxCap int
comparator Comparator<E>
LOG2 double
rand Random
maxLevel int
descendingIterator() Iterator<E>
getLast() E
add(E) boolean
computeMaxCap(int) int
logRandom() int
update(E) boolean
find(E) E
toString() String
remove(E) boolean
search(E) SLNode<E>[]
iterator() Iterator<E>
District
District()
OZGURLUK
AKSE
RESTAURANT
ATATURK
EMEK
SEKERPINAR
CAYIROVA
CUMHURIYET
INONU
values() District[]
valueOf(String) District
Graph
isEdge(int, int) boolean
isDirected() boolean
edgeIterator(int) Iterator<Edge>
getNumV() int
insert(Edge) void
getEdge(int, int) Edge
DijkstrasAlgorithm
DijkstrasAlgorithm()
dijkstrasAlgorithm(Graph, int, int[], double[]) void
AbstractGraph
AbstractGraph(int, boolean)
directed boolean
numV int
isDirected() boolean
getNumV() int
Edge
Edge(int, int)
Edge(int, int, double)
source int
dest int
weight double
equals(Object) boolean
getDest() int
toString() String
getSource() int
getWeight() double
hashCode() int
OrderStatus
OrderStatus()
ORDER_PREPARED
ORDER_DELIVERED
ORDER_TAKEN
valueOf(String) OrderStatus
values() OrderStatus[]
CustomLinkedList<E>
CustomLinkedList()
size int
head Node<E>
tail Node<E>
listIterator() ListIterator<E>
getMiddle(Node<E>) Node<E>
remove(E) E
sortedMerge(Node<E>, Node<E>) Node<E>
add(int, E) void
addLast(E) void
addFirst(E) void
get(int) E
getFirst() E
size() int
listIterator(int) ListIterator<E>
mergeSort() CustomLinkedList<E>
getLast() E
iterator() Iterator<E>
add(E) void
mergeSort(Node<E>) Node<E>
Authentication
Authentication()
scanObj Scanner
allUsers BinarySearchTree<User>
getUser(String, String) User
isPasswordTrue(String, String) boolean
getUsername() String?
getPassword(String) String?
login() User?
getUserPassword(String, String) String
isUserExist(String, String) boolean
createUser(User) void

8. SEQUENCE DIAGRAMS



9. ACTIVITY DIAGRAMS



10. NON-TRIVIAL IMPLEMENTATION DETAILS

Properties that makes the project implementation non-trivial are:

- Working as a group, rather than individually
- Concentrate on different aspects of a system as whole, rather than selecting one and work on it
- Being dependent other parts of the system, being affected from changes. This also requires stay up to date and follow updates
- Evolving to better and more reasonable way from planned at the beginning of the project.

In line with those criterias, we came up with solution methods to the project to work as a team. First of all, process is improved in github environment, to being up to date and following changes easily. Also looking back to comments shows us to evaluate the going of the project and measuring how well the initial decisions are followed, if required. Dividing the classes into manageable parts and perform related operations inside that class provide us to capability of seeing dependent parts with others, make changes easier and see relations better. The point reached as we progressed through the process showed the lack/wrongness of some initial decisions, changing the menu is one of them. And those kind of properties are

re-considered and removed if necessary.
Data structures to implement established system’s components are selected considering their prominent features.
Workers and customers are hold as ArrayList in Restaurant class due to it’s fast access by index.
To implement giveOrder, deliverOrderToCustomer LinkedList is used due to it’s traverse fast, also adding new one takes constant time and removing, editing can be fastest performed by LinkedList.
Courier’s order following system is designed as Priority Queue, in order to see the expensive order first and deliver it first.
When searching for customers from the database, binary search tree is used to perform the search process in the fastest way.

Data Structure	Reason with Complexity
List(ArrayList and LinkedList)	Workers and Customers are hold as ArrayList in Restaurant class due its fast access by index(θ(1)). Orders list,menu,representation of order are hold as Linkedlist due to fast insertion and deletion. (θ(1))
Binary Search Tree	We have a sign up / login system for Restaurant. All users must be authenticated by the system by username and password.There can be lots of user and verificate one by one take so much time so we need fast search(θ(logn) as Binary Search Tree so we used that data structure for authentication.
Queue	The chef should start preparing the first order that comest to him/her. There is no priority in the preparation of the order. Queue where first in first out(FIFO) logic used, is the most suitable data structure for this process.Also ınserting new order(offer) and removing prepared order(poll) take constant time.
Priority Queue	The courier must deliver the incoming orders according to a certain priority.We designed this priority like the most expensive order is delivered first so we used Priority Queue to give orders a priority.
TreeMap	We need to keep the ingredients of every food which in Menu. Because we need to determine if there was an igredient in the food that customers ordered which they were allergic to. Treemap key is FoodName and value of that ingredients of that food.By using TreeMap, we can access the materials of the food we want in the order in a sorted way.
Balanced AVL Tree	We made the data structure AVL tree,where we will keep the ingredients of the foods we mentioned above.Because we need to search whether there is any ingredient in the food that the customer is allergic to.By using AVL Tree, we can do this search in a fastest way(θ(logn)).

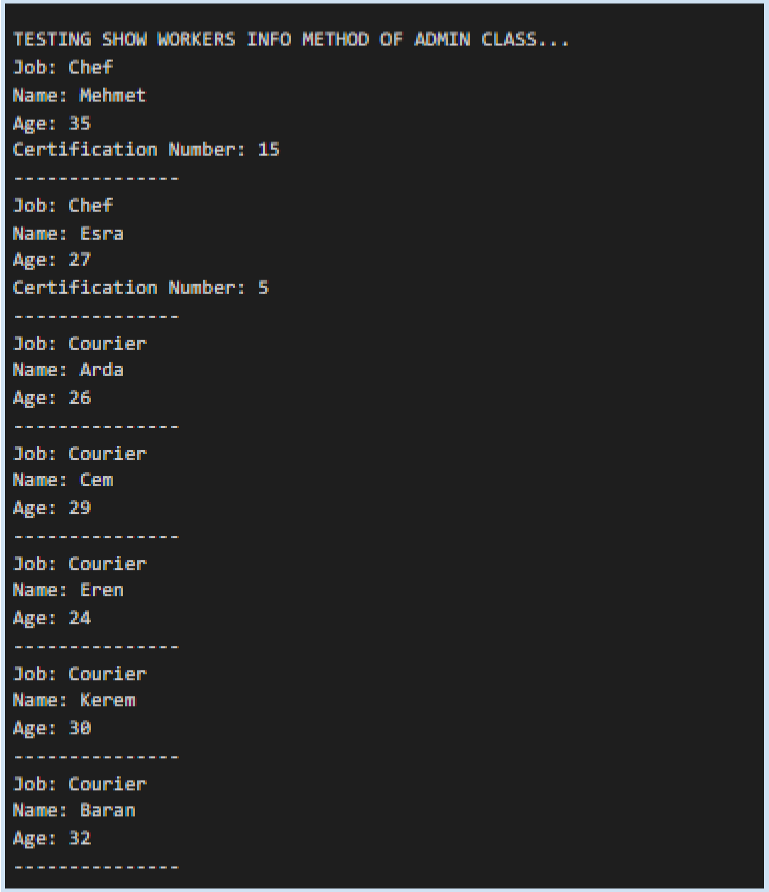
SkipList	To choose the customer of the month,we need to reach the customer with the most orders.For this reason, we must store customers in a proper order according to number of orders. We used this data structure as we were able to find the customer in a particular order or the most ordering by searching using Skiplist(Search θ(logn)).
Graph	We established a neighorhood network where the restaurant operates.The courier will also make its deliveries between these neighborhoods, but it needs to find the shortest path to make the delivery fast.We used the graph to create this network between neighboorhoods.We used the djkstra algorithm,which uses graph in the shortest path, which is our main goal.

Sorting Algorithm	Why Need sorting	Why MergeSort Choosen
MergeSort	To show the customer by sorting them according to the price amounts of the old orders of the user. Also sorting the foods in the new order given by the user according to their prices and showing them to the customer.	We need fast sort algorithm (θ(n*logn)). The first reason we chose MergeSort is that it works θ(n*logn) time complexity. Second reason, we store orders in LinkedList and MergeSort is compatible with LinkedList , does not require extra space in LinkedList

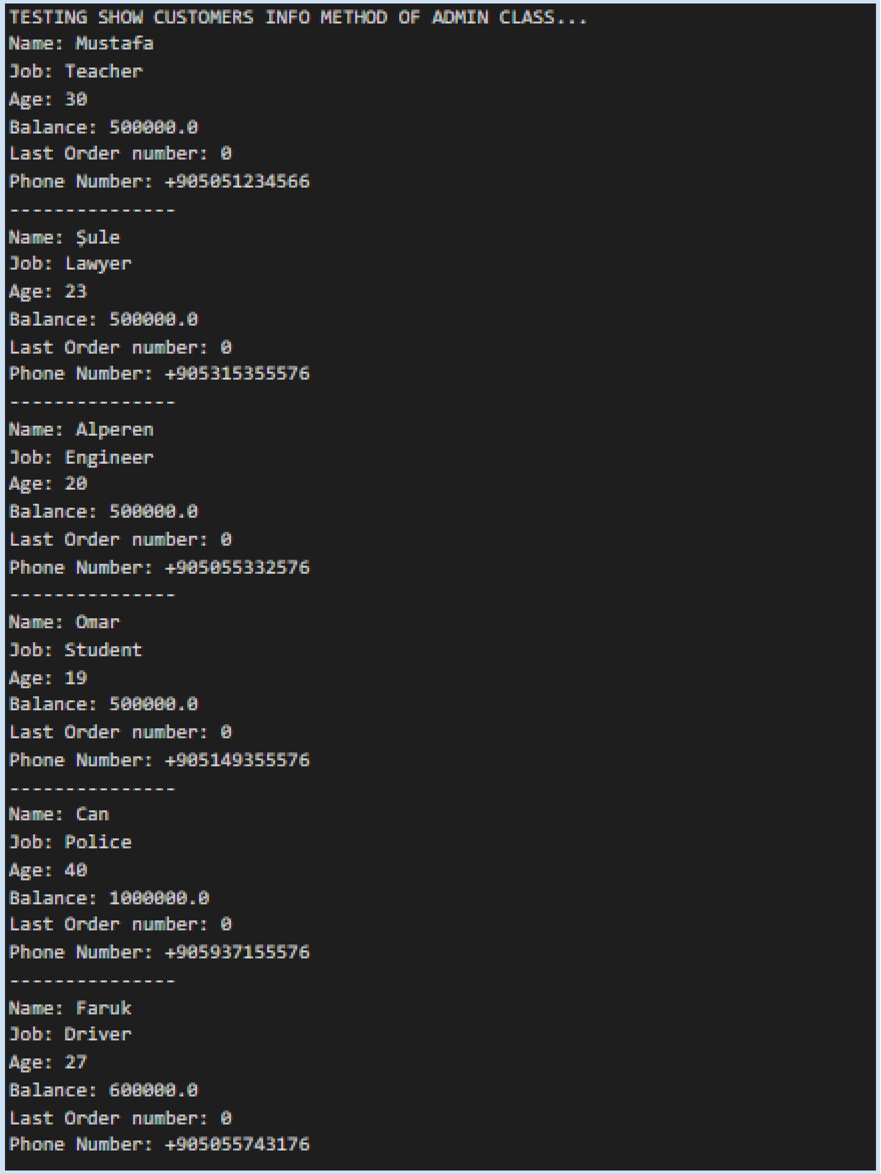
11. TEST CASES

Test ID	Test Cases	Test Steps	Test Data	Expected Result
T_Pr01	User access the system by logging.	User login the system with required inputs	Name,age,username password	User logged in the system
T_Pr02	Admin and customers see menu in the restaurant	Admin and customers see the menu by own methods	Menu	User saw the restaurants menu
T_Pr03	Admin add or delete food from menu directly	Admin creates what he /she wants and add it or delete something in menu	Food(FoodID,price FoodName,foodType)	Admin added or deleted a food in menu
T_Pr04	Admin add or delete workers from restaurant	Admin hire a selected worker or fire worst worker	Fire : - Hire: Worker	A worker fired from restaurant or hired into the restaurant
T_Pr05	Admin sees workers and customers information	Admin can see all information of workers and customers (except username-password)	List of Courier,Chef and Customer	Admin saw the information of workers and customers
T_Pr06	Admin change the salary of workers	Admin changes the salaries of the workers according to their points	Salary	Workers salary changed
T_Pr07	Customer gives the vote to workers according to order	Customer gives taste score for chef,speed score for courier	Order Score of Chef Score of Courier	Customer gived the points to the workers for order
T_Pr08	User edits own personal datas	User changes him/her personal data with new value	Name,username, password,age	User changed his/her personal information
T_Pr09	Admin sees all restaurant information	Admin sees restaurants income,outcome, score workers,customers who gave order	Restaurant	Admin saw all restaurant informations
T_Pr10	Customer send order to the restaurant	Customer select order, It sends the order list of restaurant	Order	Customer gived order and waiting for deliver
T_Pr11	Workers see order list to see the orders to be prepared	Workers see orders which will be prepared	List of Order	Workers saw the orders to be prepared
T_Ad01	Admin can see the workers info	Print the caller object of the method's job, name, age, certification info	Caller worker object	Worker's info should be seen in the terminal
T_Ad02	Admin can see the customers info	Print the caller object of the method's name, job, age, balance, last order num, phone num info	Caller customer object	Customer's info should be seen in the terminal
T_Ad03	Restaurant's income and outcome info can be seen by admin	Total income, total outcome and profit has been calculated	Restaurant object that admin's being owner	Restaurant's income, outcome, profit info should be seen in the terminal
T_Ad04	To see whether all orders are taken succesfully, admin can see all orders as a whole	In the restaurant's order field, a traverse is performed till the end	Order instances of Restaurant	All given orders should be printed to the console
T_Ad05	According to score of the worker, admin can fire the worker	All workers of the restaurant are traversed and checked for are they below rank	Workers of Restaurant	If there is a worker that has score below 4, at the end of execution of func, he/she should be fired
T_Ad06	In case of need, admin can hire new worker	To the workers field of Restaurant, an addition will be performed in given properties	Workers of Restaurant, new Worker	In case succesful adding, worker amount will increase by 1
T_Ad07	Workers' salaries can be edited by admin, depending on user votes and experience year	Worker's score, votes, current salary and experience year is evaluated and if necessary, salary will be incremented	Called Worker	If there is any change in score, vote or experience, worker's salary will be updated
T_Ad08	Admin can decide to add new food	From the scanned text, food is created with name, type, price and ID parameters	Food	New food will be prepare to be used
T_Ad09	Newly created food can be added to the menu	To the menu, a food addition will be performed	Created Food, Menu	There will be one more food on the menu after execution
T_Ad10	From the menu, a food can be deleted by admin	A remove operation will performed to delete a food from menu	Food to be delete, Menu	Given food will be removed from the menu and menu will be update
T_Ad11	A fully new menu can be prepared by admin	With constructor, a new menu will be created	New menu object	A new menu will be ready to filled with foods
T_Ad12	Admin can see the whole menu	seeMenu func will be called, that is inside menu class	Menu object	Menu will be printed to the console

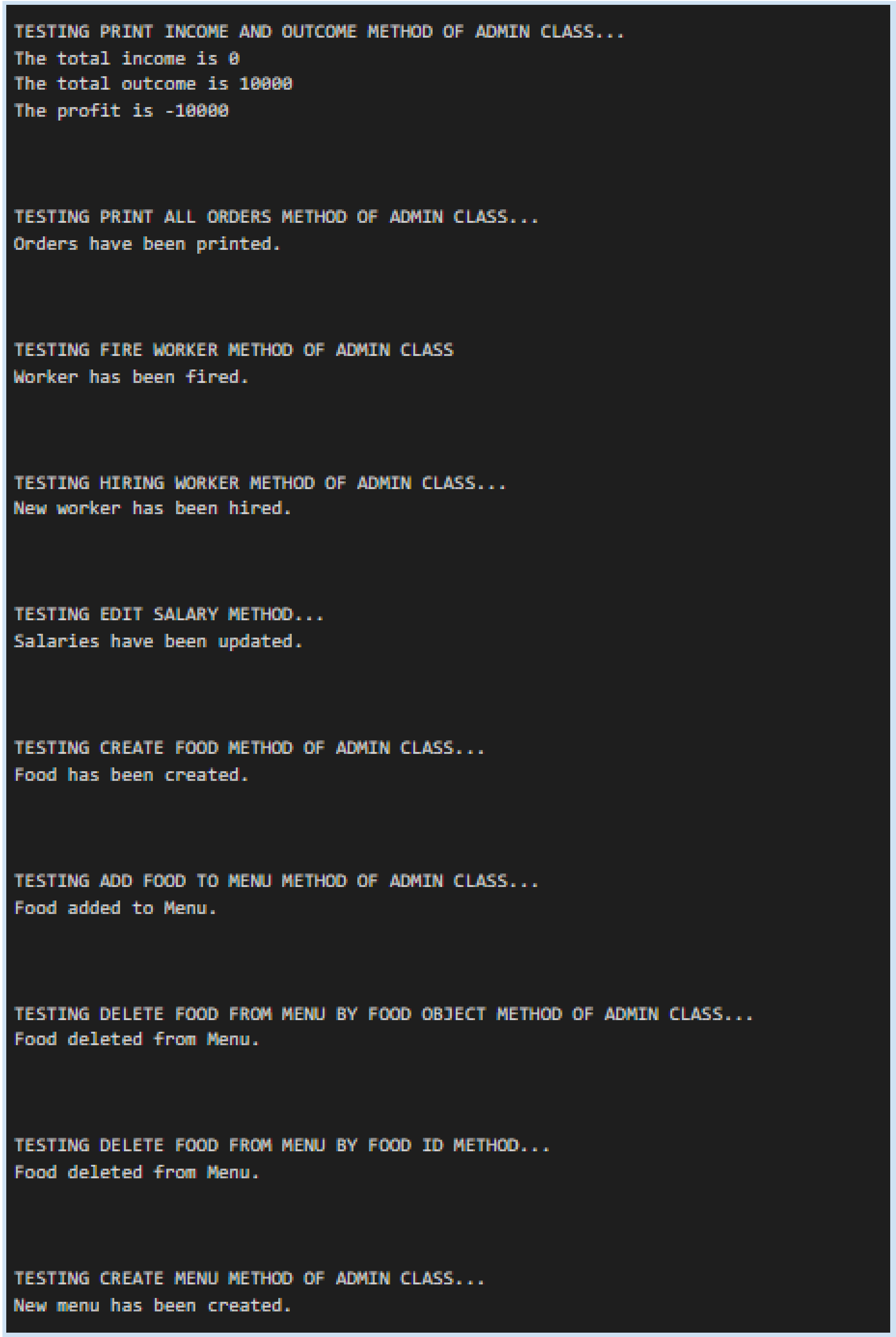
T_Ad13	Selected customer of the month can be seen by admin	Inside Restaurant class, customer of the month has been selected, holds in customers skiplist	Customers field	Last customer in SkipList will be printed to the console as customer of month
T_Ch01	Chef can add new order	To the orders field which is a Queue, a new item will be inserted	Order field of Chef	Queue's size will be increment one, in other words, a new order will be added
T_Ch02	First order will be cooked (prepared) by Chef	c	Order status	Order's status will be changed
T_Ch03	How many certificate that Chef has will be presented	Public method will return the certificate number	Certificate number field of Chef	Certificate number of Chef will be printed to the console
T_Ch04	Number of certificates of Chef can be incremented	Certificate number field will be incremented by 1	Certificate number field of Chef	Certificate number will be incremented by 1
T_Ch05	In which degree of Chef's status will be calculated	According to the successWeight, calculated with certificate number and experience year, stat of chef will be calculated	Certificate number, experience year fields of Chef	In case any major change in fields certificate number and experience year, stat of Chef will be updated
T_Ch06	Order's size can be seen by Chef	Public method will return the order (queue) size	Order field of Chef	Order size of Chef will be printed to the console
T_Ch07	Two chef's can be compare with each other	Two object's identity will be compared	Two Chef object	If they're same person, comparison will return true
T_Co01	Courier's phone number can be update	Old phone number will update to the new one	Phone number field of Courier	Courier's phone number will be change with updated one
T_Co02	Courier's phone number can be seen	Current phone num of courier can be seen	Phone number field of Courier	Courier's phone number will be print to the console
T_Co03	Order's size can be seen by Courier	Public method will return the order (PriorityQueue) size	Order field of Courier	Order size of Courier will be printed to the console
T_Co04	Courier can add order to the him/her own field, which is PriorityQueue	To the orders field which is a PriorityQueue, a new item will be inserted	Order field of Courier	Order's size will be increment one, in other words, a new order will be added
T_Co05	Order deliver to the customer performed	Order's status will be set as "delivered"	Order status	Order's status will be changed
T_Co06	Value of districts can be calculated by Courier	District values will be find	District	We'll know the district values
T_Co07	Shortest route for districts can be find by Courier	Shortest path algorithm will be performed	District	Shortest route will be find
T_Co08	Two couriers can be compare with each other	Two object's identity will be compared	Two Courier object	If they're same person, comparison will return true
T_Co09	Shortest route for districts can be seen by Courier	Shortest path algorithm will be shown	District	Shortest route will be presented
T_Cu01	Perform login to the system as customer	With the constructor, according to readed data, customer login will performed	Customer dataset	In case login is succesfull, a new customer will be in the system
T_Cu02	Give order as customer	To the existing queue, current customer will contriburete	Worker's queue field	An order will be recorded
T_Cu03	See the menu as customer	Menu class will be presented	Menu	Menu of restaurant will be printed to the console



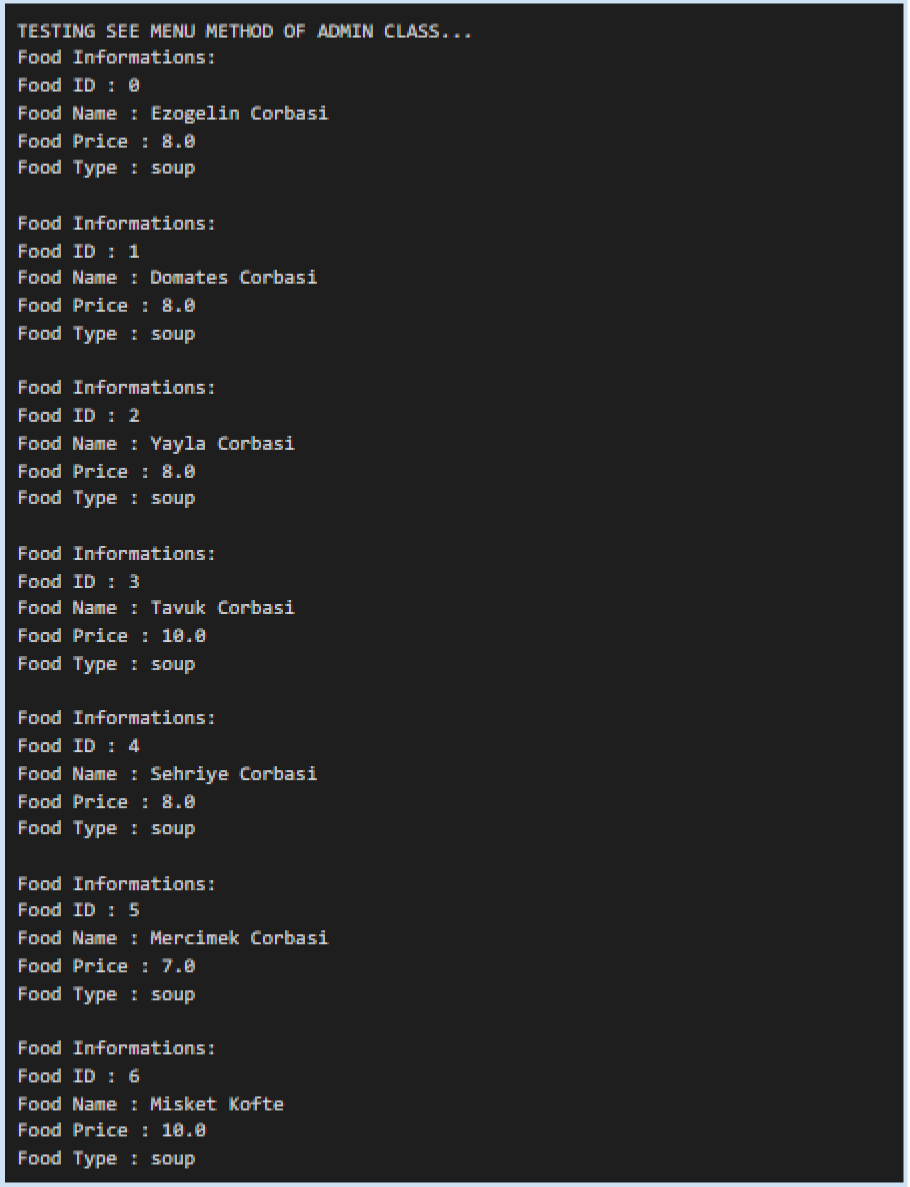
```
TESTING SHOW WORKERS INFO METHOD OF ADMIN CLASS...
Job: Chef
Name: Mehmet
Age: 35
Certification Number: 15
---------------
Job: Chef
Name: Esra
Age: 27
Certification Number: 5
---------------
Job: Courier
Name: Arda
Age: 26
---------------
Job: Courier
Name: Cem
Age: 29
---------------
Job: Courier
Name: Eren
Age: 24
---------------
Job: Courier
Name: Kerem
Age: 30
---------------
Job: Courier
Name: Baran
Age: 32
---------------
```



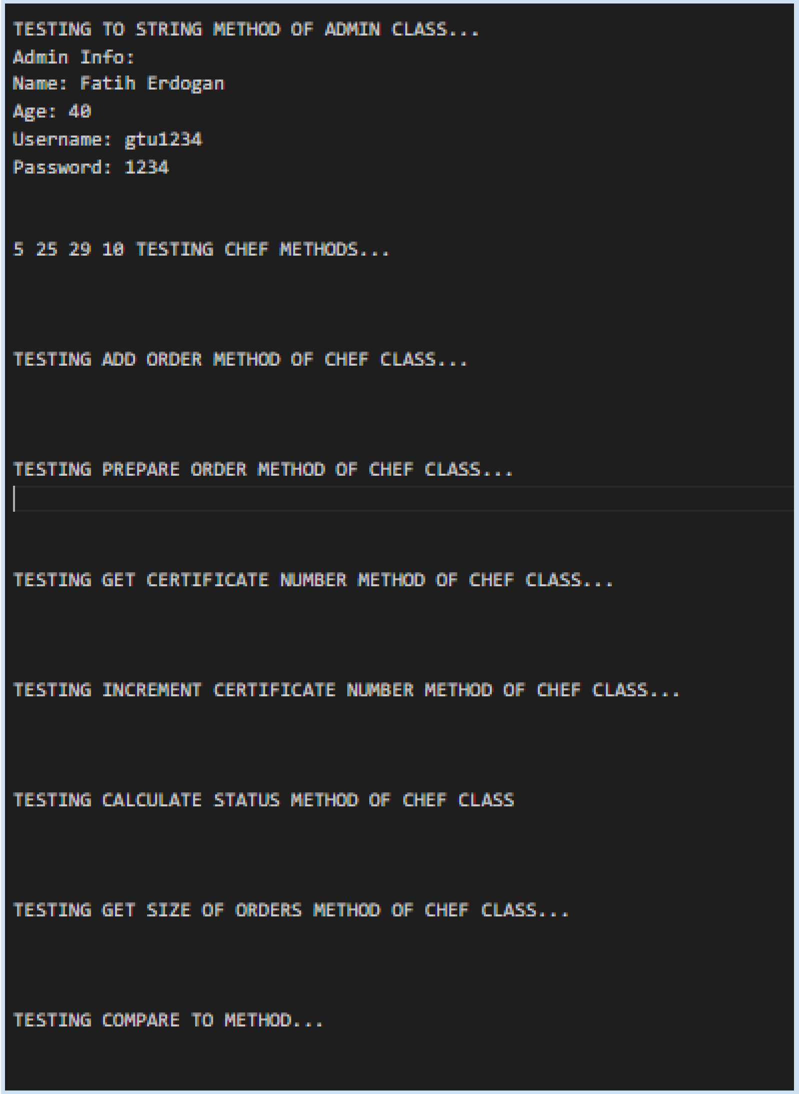
```
TESTING SHOW CUSTOMERS INFO METHOD OF ADMIN CLASS...
Name: Mustafa
Job: Teacher
Age: 30
Balance: 500000.0
Last Order number: 0
Phone Number: +905051234566
---------------
Name: Şule
Job: Lawyer
Age: 23
Balance: 500000.0
Last Order number: 0
Phone Number: +905315355576
---------------
Name: Alperen
Job: Engineer
Age: 20
Balance: 500000.0
Last Order number: 0
Phone Number: +905055332576
---------------
Name: Omar
Job: Student
Age: 19
Balance: 500000.0
Last Order number: 0
Phone Number: +905149355576
---------------
Name: Can
Job: Police
Age: 40
Balance: 1000000.0
Last Order number: 0
Phone Number: +905937155576
---------------
Name: Faruk
Job: Driver
Age: 27
Balance: 600000.0
Last Order number: 0
Phone Number: +905055743176
```



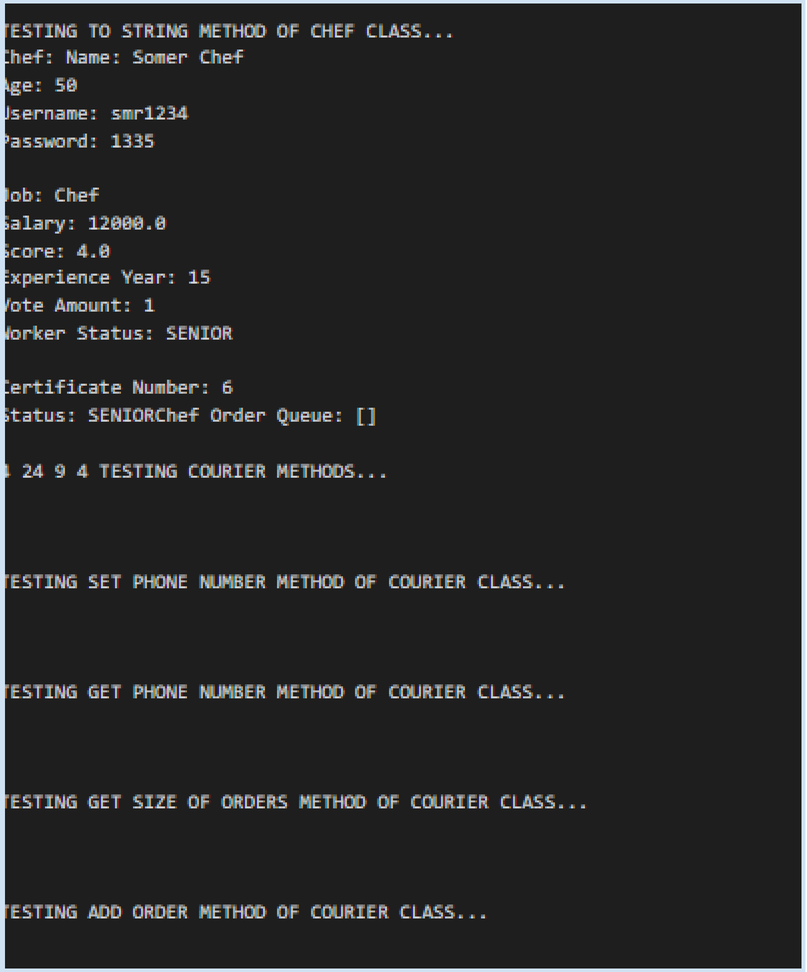
```
TESTING PRINT INCOME AND OUTCOME METHOD OF ADMIN CLASS...
The total income is 0
The total outcome is 10000
The profit is -10000



TESTING PRINT ALL ORDERS METHOD OF ADMIN CLASS...
Orders have been printed.



TESTING FIRE WORKER METHOD OF ADMIN CLASS
Worker has been fired.



TESTING HIRING WORKER METHOD OF ADMIN CLASS...
New worker has been hired.



TESTING EDIT SALARY METHOD...
Salaries have been updated.



TESTING CREATE FOOD METHOD OF ADMIN CLASS...
Food has been created.



TESTING ADD FOOD TO MENU METHOD OF ADMIN CLASS...
Food added to Menu.



TESTING DELETE FOOD FROM MENU BY FOOD OBJECT METHOD OF ADMIN CLASS...
Food deleted from Menu.



TESTING DELETE FOOD FROM MENU BY FOOD ID METHOD...
Food deleted from Menu.



TESTING CREATE MENU METHOD OF ADMIN CLASS...
New menu has been created.
```



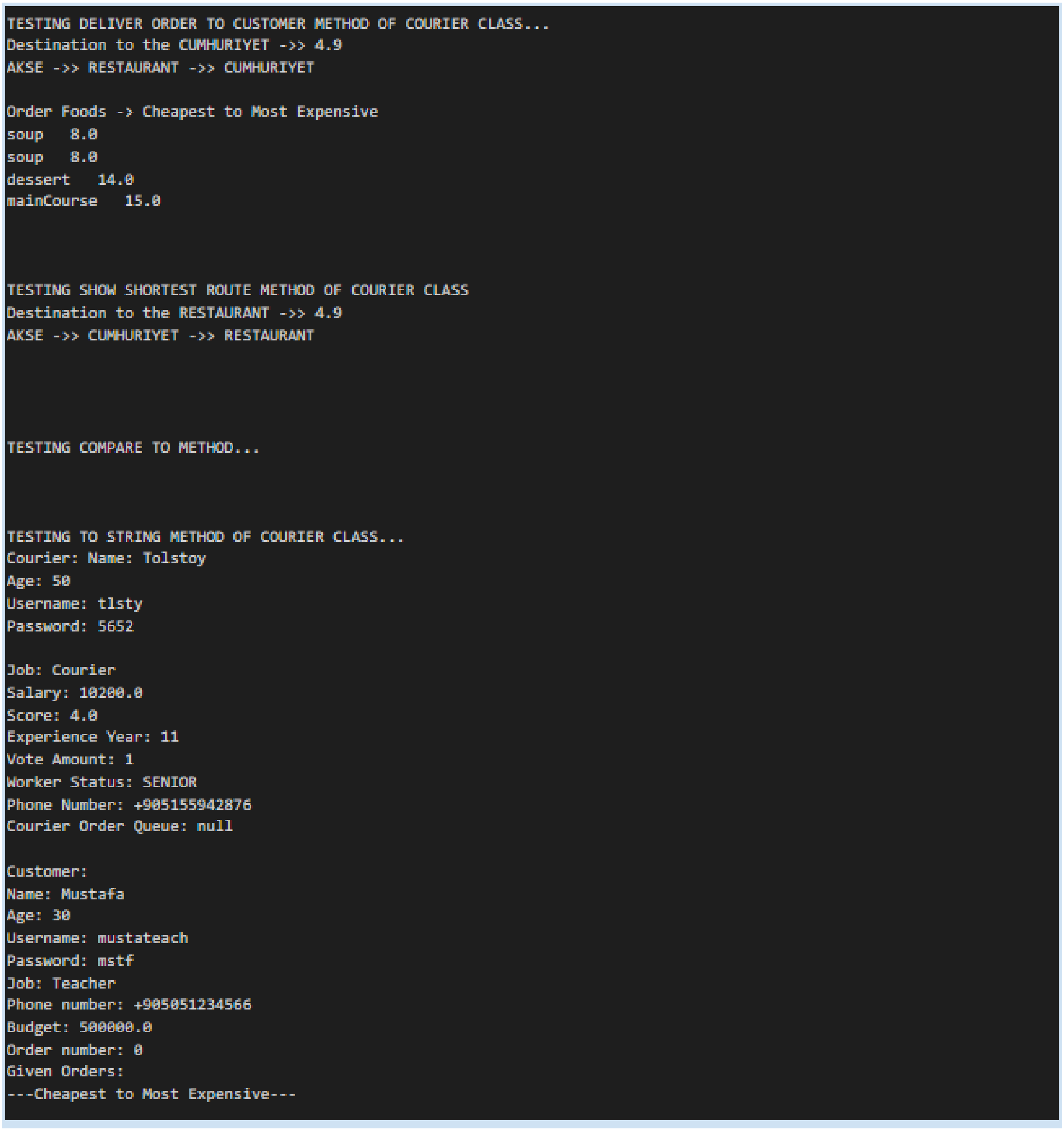
```
TESTING SEE MENU METHOD OF ADMIN CLASS...
Food Informations:
Food ID : 0
Food Name : Ezogelin Corbasi
Food Price : 8.0
Food Type : soup

Food Informations:
Food ID : 1
Food Name : Domates Corbasi
Food Price : 8.0
Food Type : soup

Food Informations:
Food ID : 2
Food Name : Yayla Corbasi
Food Price : 8.0
Food Type : soup

Food Informations:
Food ID : 3
Food Name : Tavuk Corbasi
Food Price : 10.0
Food Type : soup

Food Informations:
Food ID : 4
Food Name : Sehriye Corbasi
Food Price : 8.0
Food Type : soup

Food Informations:
Food ID : 5
Food Name : Mercimek Corbasi
Food Price : 7.0
Food Type : soup

Food Informations:
Food ID : 6
Food Name : Misket Kofte
Food Price : 10.0
Food Type : soup
```



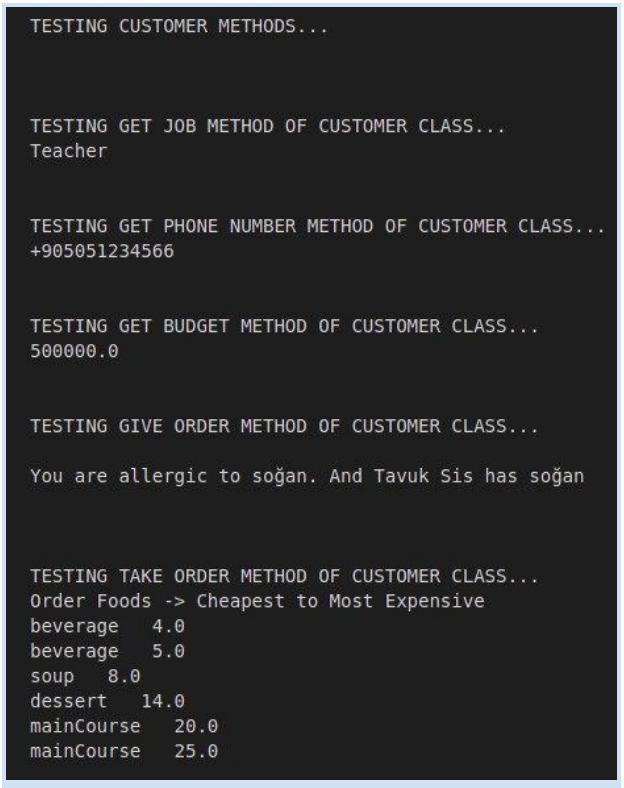
```
TESTING TO STRING METHOD OF ADMIN CLASS...
Admin Info:
Name: Fatih Erdogan
Age: 40
Username: gtu1234
Password: 1234


5 25 29 10 TESTING CHEF METHODS...



TESTING ADD ORDER METHOD OF CHEF CLASS...



TESTING PREPARE ORDER METHOD OF CHEF CLASS...




TESTING GET CERTIFICATE NUMBER METHOD OF CHEF CLASS...



TESTING INCREMENT CERTIFICATE NUMBER METHOD OF CHEF CLASS...



TESTING CALCULATE STATUS METHOD OF CHEF CLASS



TESTING GET SIZE OF ORDERS METHOD OF CHEF CLASS...



TESTING COMPARE TO METHOD...
```



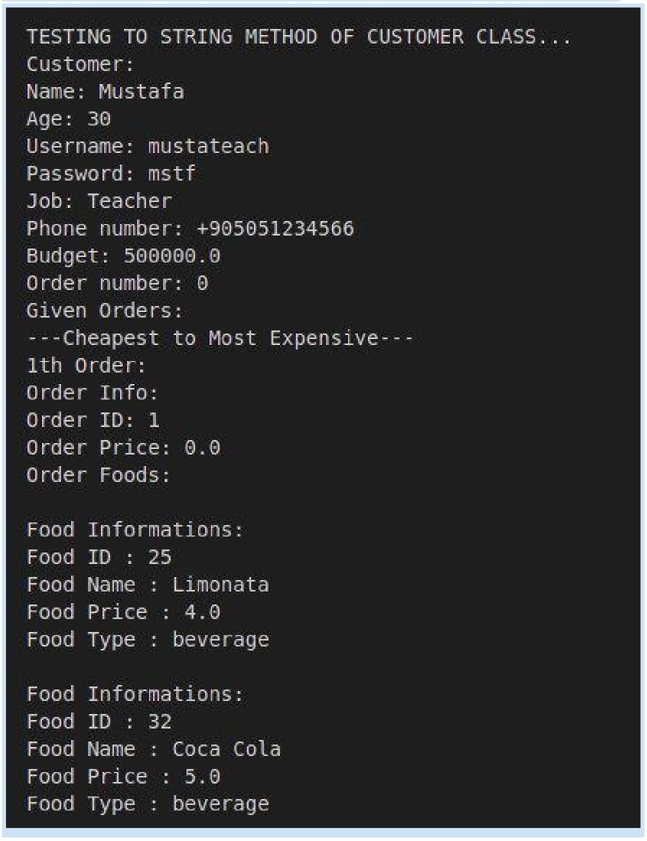
```
ESTING TO STRING METHOD OF CHEF CLASS...
hef: Name: Somer Chef
ge: 50
sername: smr1234
assword: 1335

ob: Chef
alary: 12000.0
core: 4.0
xperience Year: 15
ote Amount: 1
orker Status: SENIOR

ertificate Number: 6
tatus: SENIORChef Order Queue: []

 24 9 4 TESTING COURIER METHODS...



ESTING SET PHONE NUMBER METHOD OF COURIER CLASS...



ESTING GET PHONE NUMBER METHOD OF COURIER CLASS...



ESTING GET SIZE OF ORDERS METHOD OF COURIER CLASS...



ESTING ADD ORDER METHOD OF COURIER CLASS...
```



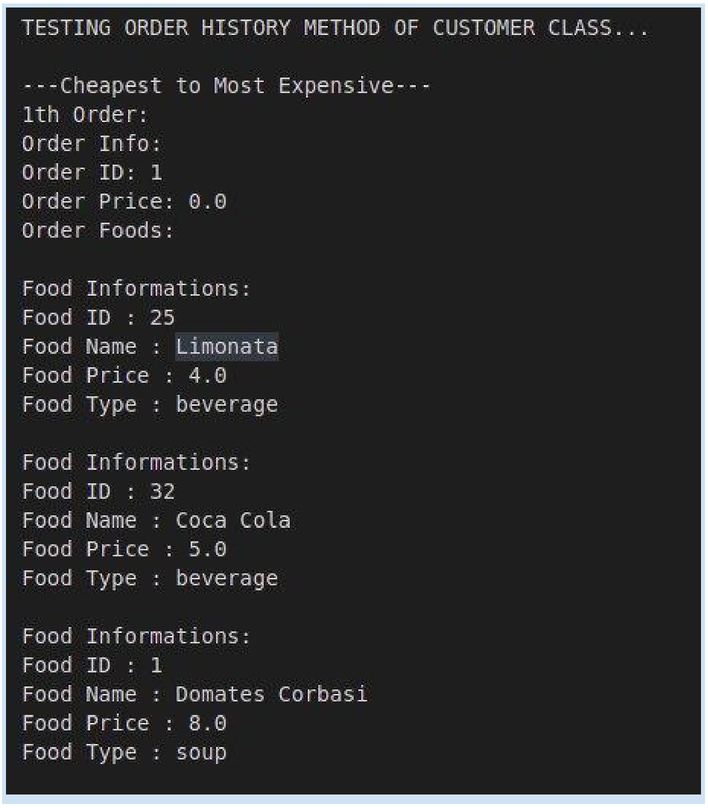
```
TESTING DELIVER ORDER TO CUSTOMER METHOD OF COURIER CLASS...
Destination to the CUMHURIYET ->> 4.9
AKSE ->> RESTAURANT ->> CUMHURIYET

Order Foods -> Cheapest to Most Expensive
soup   8.0
soup   8.0
dessert   14.0
mainCourse   15.0



TESTING SHOW SHORTEST ROUTE METHOD OF COURIER CLASS
Destination to the RESTAURANT ->> 4.9
AKSE ->> CUMHURIYET ->> RESTAURANT




TESTING COMPARE TO METHOD...



TESTING TO STRING METHOD OF COURIER CLASS...
Courier: Name: Tolstoy
Age: 50
Username: tlsty
Password: 5652

Job: Courier
Salary: 10200.0
Score: 4.0
Experience Year: 11
Vote Amount: 1
Worker Status: SENIOR
Phone Number: +905155942876
Courier Order Queue: null

Customer:
Name: Mustafa
Age: 30
Username: mustateach
Password: mstf
Job: Teacher
Phone number: +905051234566
Budget: 500000.0
Order number: 0
Given Orders:
---Cheapest to Most Expensive---
```



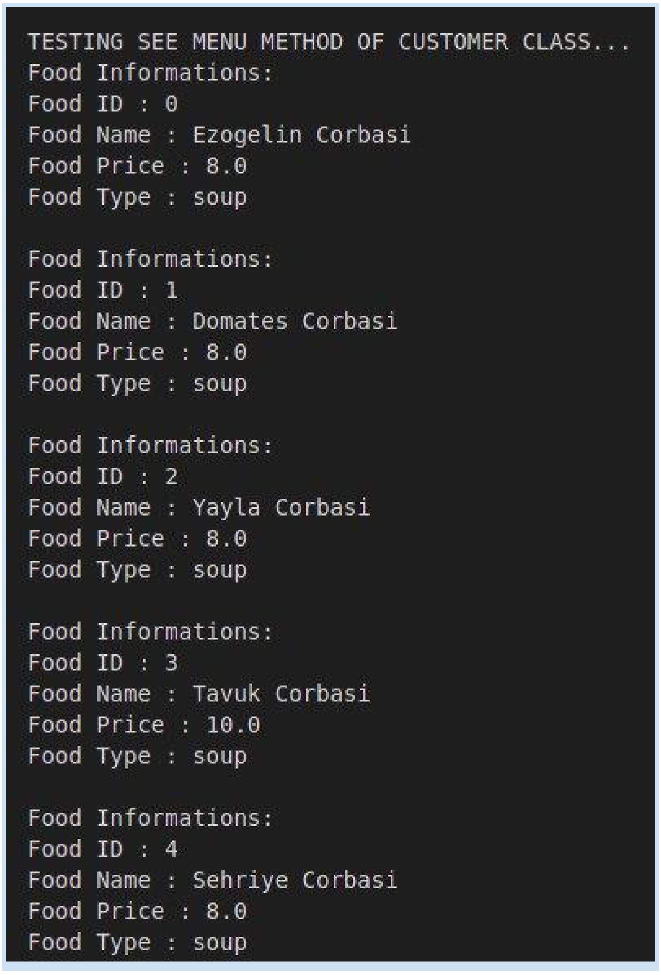
```
TESTING CUSTOMER METHODS...



TESTING GET JOB METHOD OF CUSTOMER CLASS...
Teacher


TESTING GET PHONE NUMBER METHOD OF CUSTOMER CLASS...
+905051234566


TESTING GET BUDGET METHOD OF CUSTOMER CLASS...
500000.0


TESTING GIVE ORDER METHOD OF CUSTOMER CLASS...

You are allergic to soğan. And Tavuk Sis has soğan



TESTING TAKE ORDER METHOD OF CUSTOMER CLASS...
Order Foods -> Cheapest to Most Expensive
beverage   4.0
beverage   5.0
soup   8.0
dessert   14.0
mainCourse   20.0
mainCourse   25.0
```



```
TESTING TO STRING METHOD OF CUSTOMER CLASS...
Customer:
Name: Mustafa
Age: 30
Username: mustateach
Password: mstf
Job: Teacher
Phone number: +905051234566
Budget: 500000.0
Order number: 0
Given Orders:
---Cheapest to Most Expensive---
1th Order:
Order Info:
Order ID: 1
Order Price: 0.0
Order Foods:

Food Informations:
Food ID : 25
Food Name : Limonata
Food Price : 4.0
Food Type : beverage

Food Informations:
Food ID : 32
Food Name : Coca Cola
Food Price : 5.0
Food Type : beverage
```



```
TESTING ORDER HISTORY METHOD OF CUSTOMER CLASS...

---Cheapest to Most Expensive---
1th Order:
Order Info:
Order ID: 1
Order Price: 0.0
Order Foods:

Food Informations:
Food ID : 25
Food Name : Limonata
Food Price : 4.0
Food Type : beverage

Food Informations:
Food ID : 32
Food Name : Coca Cola
Food Price : 5.0
Food Type : beverage

Food Informations:
Food ID : 1
Food Name : Domates Corbasi
Food Price : 8.0
Food Type : soup
```



```
TESTING SEE MENU METHOD OF CUSTOMER CLASS...
Food Informations:
Food ID : 0
Food Name : Ezogelin Corbasi
Food Price : 8.0
Food Type : soup

Food Informations:
Food ID : 1
Food Name : Domates Corbasi
Food Price : 8.0
Food Type : soup

Food Informations:
Food ID : 2
Food Name : Yayla Corbasi
Food Price : 8.0
Food Type : soup

Food Informations:
Food ID : 3
Food Name : Tavuk Corbasi
Food Price : 10.0
Food Type : soup

Food Informations:
Food ID : 4
Food Name : Sehriye Corbasi
Food Price : 8.0
Food Type : soup
```

12. PERFORMANCE ANALYSIS

a. Theoretical Analysis

Methods that work at constant time,which we did not write an explanation for,print something to inform the user or stable mathematical operations independent of the number of elements.

Worker Class

METHOD NAME	COMPLEXITY
calculateSalary	Θ(1)
calculateAverageScore	Θ(1)
dismissalControl	Θ(1)

Admin Class

METHOD NAME	COMPLEXITY
ShowWorkersInfo	Θ(n)

constant time print operation for n element(Worker List) with loop

ShowCustomersInfo	Θ(n)

constant time print operation for n element (Worker List) with loop

printIncomeAndOutcome	Θ(1)
printAllOrders	Θ(n)

constant time print operation for n element (Order List) with loop

fireWorker	Θ(n)

controlling n worker,if score is low remove it

hiringWorker	Θ(1)

add Arraylist new element take constant time

editSalary	**Θ(n)**

constant time setting salary operation for n element(worker List)

createFood	**Θ(1)**

Menu Class

METHOD NAME	**COMPLEXITY**
seeMenu	**Θ(n)**

constant time print operation for n element

Customer Class

METHOD NAME	**COMPLEXITY**
myOrders	**Θ(n*logn)**

Doing merge sort for n element(Order List) it takes n*logn time + constant time print operation for n element = θ(n*logn)

orderHistory	**Θ(n*logn)**
giveVote	**Θ(1)**
giveOrder	**Θ(1)**
takeOrder	**Θ(n*logn)**

Doing merge sort for n element(Food list) it takes n*logn time + constant time print operation for n element = θ(n*logn)

isVIP/isStudent	**Θ(1)**
checkAllergy	**O(n*n*logn)**

Search all order which is n and in this loop search all allergies of customer which is n and check whether the allergy of customer exists.

Chef Class

METHOD NAME	**COMPLEXITY**

addOrder	**Θ(1)**

queue offer/add operation takes constant time

prepareOrder	**Θ(1)**

queue poll/remove operation takes constant time

getCertificateNumber	**Θ(1)**
incrementCertificateNumber	**Θ(1)**
calculateStatus	**Θ(1)**
getSizeOfOrders	**Θ(1)**
toString	**Θ(1)**

User Class

METHOD NAME	**COMPLEXITY**
just setters/getters	**Θ(1)**

Courier Class

METHOD NAME	**COMPLEXITY**
addOrder	**Θ(log(n))**

Priority queue offer/add operation takes logn time.

showShortestRoute	**Θ(e*logv)**
deliverOrderToCustomer	**Θ(e*logv)**

e is the total number of edges and v is the total number of vertices.

setPhoneNumber	**Θ(1)**

getPhoneNumber	Θ(1)
calculateStatus	Θ(1)
getSizeOfOrders	Θ(1)
toString	Θ(1)

Authentication Class

METHOD NAME	COMPLEXITY
logIn	Θ(n)
createUser	Θ(log(n))
getMenuFromDatabase	Θ(n)
getCustomersFromDatabase	Θ(n)
getAllUsersFromDatabase	Θ(n.log(n))
getUsernameFromUserForLogIn	Θ(log(n))
getPasswordFromUserForLogIn	Θ(n)
isUserExist	Θ(log(n))
isPasswordTrue	Θ(n)
getUserPassword	Θ(log(n))
getUserFromUsername	Θ(log(n))
parseFoodLine	Θ(n)
parseAndConvertUserLine	Θ(n)
parseUserLine	Θ(n)

Order Class

METHOD NAME	COMPLEXITY
coefCalc	Θ(1)
calculateAccount	O(n)

Constant time logical operation for n element(Food list) with loop

toString	O(n)
Compare	Θ(1)

Food Class

METHOD NAME	COMPLEXITY
toString	Θ(n)

Restaurant Class

METHOD NAME	COMPLEXITY
createRandomFoods	Θ(1)
calculateScore	Θ(n)

Constant time logical operation for n element(Worker List) with loop

addOrder	Θ(1)

LinkedList add operation takes constant time

chooseChef	O(n)

Choose a chef which has minimum order with looking all n element workers

chooseCourier	O(n)

Choose a courier which has minimum order with looking all n element workers

showWorkers	Θ(n)

Constant time print operation for n element(Worker list)

menu	Θ(n)

Constant time print operation for n element(Food list)

deleteFoodFromMenu(int) or (food)	**O(n)**

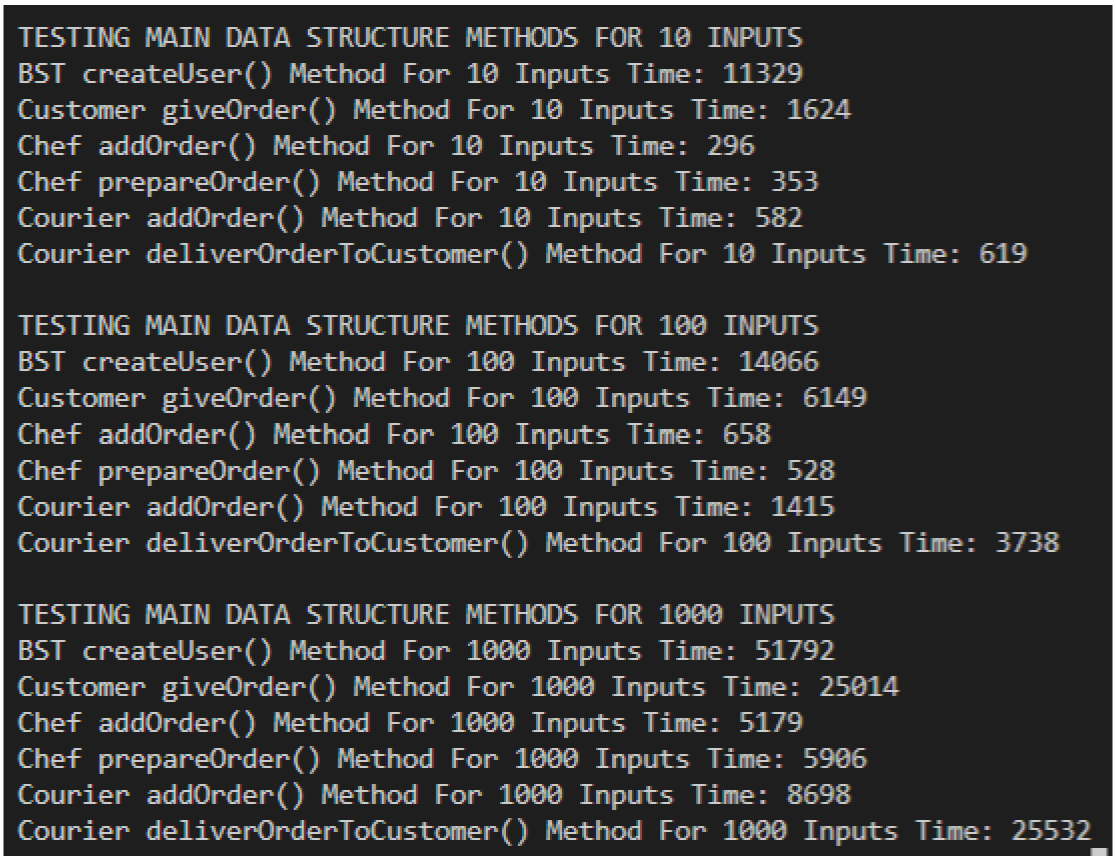
Searching n element LinkedList and if find target element remove it(takes constant time)

addFoodtoMenu	**Θ(1)**

Adding Linkedlist takes constant time

createNewMenu	**Θ(1)**
toString	**Θ(n)**

b. Experimental Analysis



```
TESTING MAIN DATA STRUCTURE METHODS FOR 10 INPUTS
BST createUser() Method For 10 Inputs Time: 11329
Customer giveOrder() Method For 10 Inputs Time: 1624
Chef addOrder() Method For 10 Inputs Time: 296
Chef prepareOrder() Method For 10 Inputs Time: 353
Courier addOrder() Method For 10 Inputs Time: 582
Courier deliverOrderToCustomer() Method For 10 Inputs Time: 619

TESTING MAIN DATA STRUCTURE METHODS FOR 100 INPUTS
BST createUser() Method For 100 Inputs Time: 14066
Customer giveOrder() Method For 100 Inputs Time: 6149
Chef addOrder() Method For 100 Inputs Time: 658
Chef prepareOrder() Method For 100 Inputs Time: 528
Courier addOrder() Method For 100 Inputs Time: 1415
Courier deliverOrderToCustomer() Method For 100 Inputs Time: 3738

TESTING MAIN DATA STRUCTURE METHODS FOR 1000 INPUTS
BST createUser() Method For 1000 Inputs Time: 51792
Customer giveOrder() Method For 1000 Inputs Time: 25014
Chef addOrder() Method For 1000 Inputs Time: 5179
Chef prepareOrder() Method For 1000 Inputs Time: 5906
Courier addOrder() Method For 1000 Inputs Time: 8698
Courier deliverOrderToCustomer() Method For 1000 Inputs Time: 25532
```