Restaurant Food Ordering and Table Booking System

https://github.com/bangrui001/OOP_Final_Project_25Spring

Group 10
Bangrui Yan
Yaoge Hu
YiCi Zhang

Date of Submission: 05/10/2025

Table of Work

(Please write x in the boxes to mention what each student achieved in this project)

	Bangrui Yan	Yaoge Hu	YiCi Zhang
Project Description	x	x	x
Uses Cases Diagram(s)	x	x	x
Sequence Diagrams	x	x	x
Class diagram(s)	x	x	x
Implementation	x		x
Conclusion	x	x	x

General Description:

The Restaurant Ordering and Seat Booking System is a console-based Java application designed for small to medium-sized restaurants to digitize and streamline their food ordering and table reservation processes. The system distinguishes between Admin and Customer roles, providing each with a tailored set of features such as menu management, food ordering, and seat reservations. Built using Object-Oriented Programming principles, it offers modularity and clarity.

Main Features of our System

1: Admin Functionalities

- Add, delete, and edit food items in the menu.
- View details of all menu items.
- View food items ordered by specific customers.
- View the list of customers who ordered a specific dish.
- Manage table availability.
- View all table reservations.

2: Customer Functionalities

- Create a user profile.
- View menu and order food items.
- Cancel food items from orders.
- View and calculate total order price.
- Book and cancel table reservations.
- View personal reservation history.

Our System Architecture:

The system is built following Object-Oriented Programming (OOP) principles with the following structure:

- `RestaurantApp`: Main application controller with `main()` method.
- `Admin`, `Customer`: User roles inheriting from a base `User` class.
- `SystemDirectory`: Service layer for managing food entries, customer data, and seat operations.
- `FoodEntry`, `Seat`, `SeatReservation`: Core domain entities.
- Interfaces: `UserInterface`, `AdminInterface`, `customerInterface` for role-specific behaviors.

All operations are performed **in-memory** using `ArrayList` structures.

Goals and Benefits:

Our first goal is to provide an intuitive interface for restaurant customers to order food and reserve tables. Our second goal is to enable administrators to efficiently manage the restaurant's menu and table availability. The third goal is to do some exercise on the OOP design.

The first benefit is that our program put the shared fields and methods of customer and admin class in the user class. then, we extend the customer and admin to user, which reduce the code repetition.

The second benefit is that it can reduce manual effort and potential human errors in food ordering and table booking since we use program.

the special requirement is the default username is admin. user are not allowed to change the user name of the admin.

System Input(s) and Output(s):

Inputs:
The system accepts user input through command-line interactions. Depending on the role (Admin or Customer), inputs include:

- User Role Selection: Choosing between Admin, Customer, or Exit.
- Authentication Input: Admin username verification.
- Menu Management Inputs (Admin):
  - Food name, ID, price, and category for creating or editing menu items.
  - Food name for deletion or lookup.
- Order and Profile Inputs (Customer):
  - Customer username, phone, and email for profile creation.
  - Food item name for ordering or canceling.
- Seat Reservation Inputs:
  - Party size and time slot for table booking.
  - Table ID and time for reservation or cancellation.

All inputs are received via Scanner objects reading from the console.

Outputs:
The system provides textual outputs via the console to reflect operation results, status updates, and data listings:

- Feedback Messages:
  - Success or failure of actions (e.g., order placed, table reserved).
- Menu Displays:
  - Full menu with food details (name, ID, price, category).
- Customer Order Summary:
  - List of ordered items and total price.
- Seat Booking Information:
  - List of available tables, reservation confirmations, and cancellations.
- Admin Management Reports:
  - Lists of customers who ordered specific items, or foods ordered by specific customers.

Some Special Requirement:

Food Ordering System
1: The default username for the Admin is Admin (case-insensitive).
2: The system comes with a predefined menu of ten dishes, covering various categories like Entrée, Appetizer, Dessert, and Beverage.
3: Food IDs should be entered as integers like 5.

Table Booking System
1: The reservation system is designed for same day use only. Customers and Admins can only make or view reservations for the current day. Advance reservations (e.g., for tomorrow) are not supported.
2: Each table can be reserved for 1-hour time slots starting only at whole-hour marks (e.g., 18:00, 19:00), and the User should enter integer to indicate time like 18 for 6:00 PM.
3: Table IDs start with the letter T (e.g., T5). Both Admin and Customer must input the correct table ID format when making or managing reservations.

Use Cases Diagram and Use Cases Description

UC Reference Name/Number: 1	
Overview	Add a new food item to the restaurant's food menu.
Related use cases:	
Actors	Admin

UC Reference Name/Number: 2	
Overview	Remove a food item from the restaurant's food menu.
Related use cases:	
Actors	Admin

UC Reference Name/Number: 3	
Overview	Edit the details of a food item including price and category on the restaurant's food menu
Related use cases:	
Actors	Admin

UC Reference Name/Number: 4	
Overview	Display information about name, id, price and dish type for a given Food item.
Related use cases:	
Actors	Admin, Customer

UC Reference Name/Number: 5	
Overview	View all the food items' information on the food menu.
Related use cases:	
Actors	Admin, Customer

UC Reference Name/Number: 6	
Overview	View the List of Food items ordered by a given Customer
Related use cases:	
Actors	Admin

UC Reference Name/Number: 7	
Overview	Views list of customers ordered a specific food.
Related use cases:	
Actors	Admin

UC Reference Name/Number: 8	
Overview	Create a User Profile for using the system
Related use cases:	
Actors	Customer

UC Reference Name/Number: 9	
Overview	Order the food by Food Name
Related use cases:	
Actors	Customer

UC Reference Name/Number: 10	
Overview	Cancel a ordered food by Food Name
Related use cases:	
Actors	Customer

UC Reference Name/Number: 11	
Overview	Get the total price of ordered foods
Related use cases:	
Actors	Customer

UC Reference Name/Number: 12	
Overview	Views all the Table reservations today
Related use cases:	
Actors	Admin

UC Reference Name/Number:13	
Overview	Manages which Table is ready to reserve now
Related use cases:	
Actors	Admin

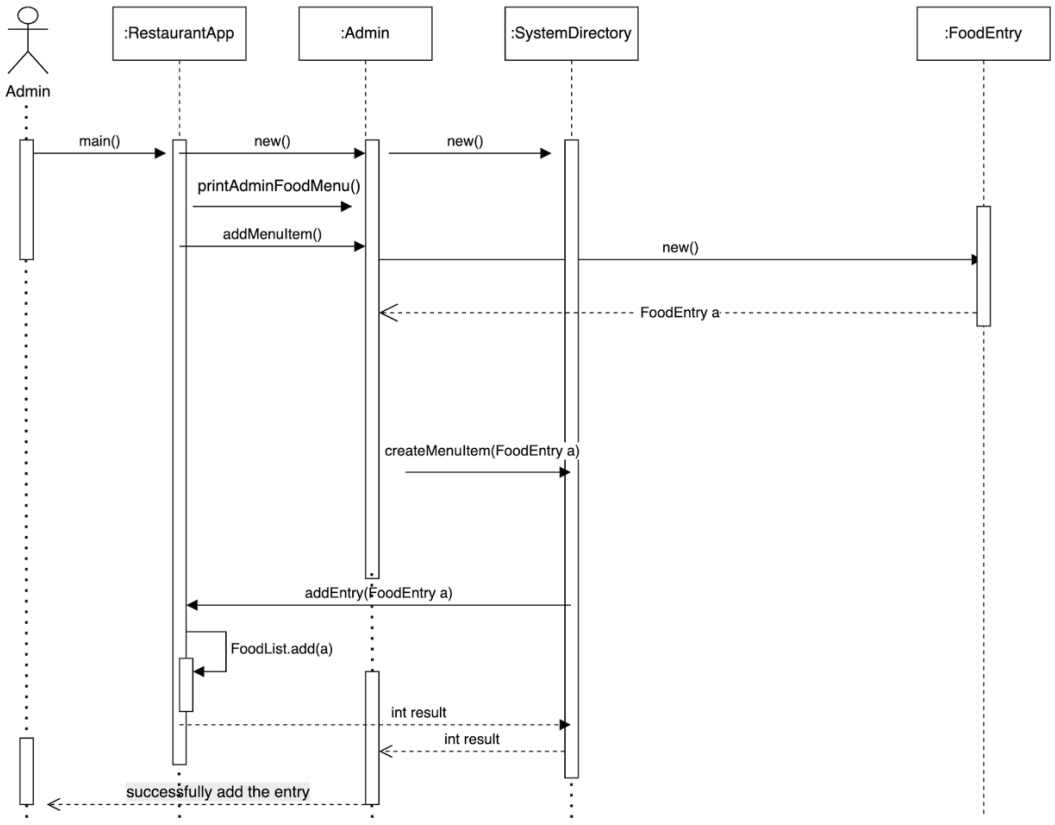
UC Reference Name/Number:14	
Overview	Book a table based on time and number of people
Related use cases:	
Actors	Customer

UC Reference Name/Number: 15	
Overview	Cancel a booked table
Related use cases:	
Actors	Customer

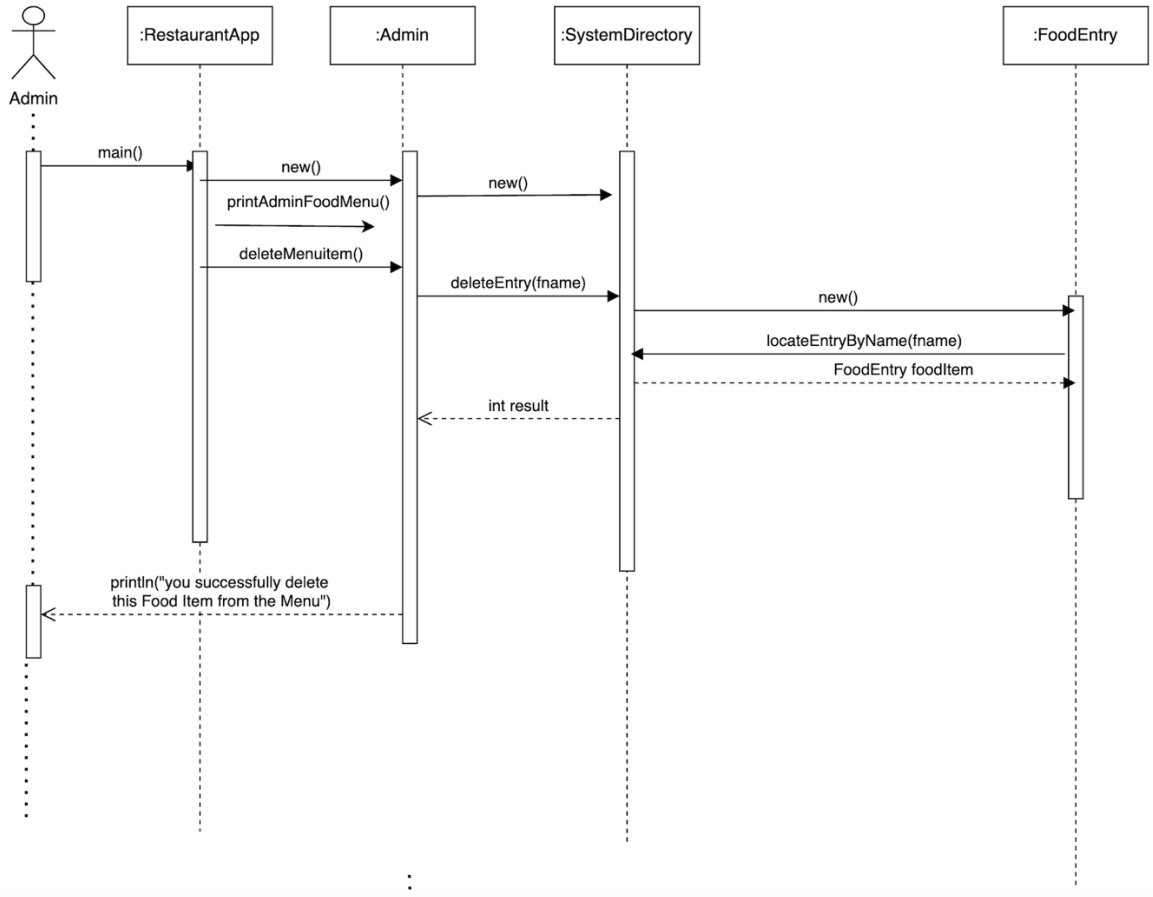
UC Reference Name/Number: 16	
Overview	Views information of the booked tables
Related use cases:	
Actors	Customer

System Design:
Sequence Diagram

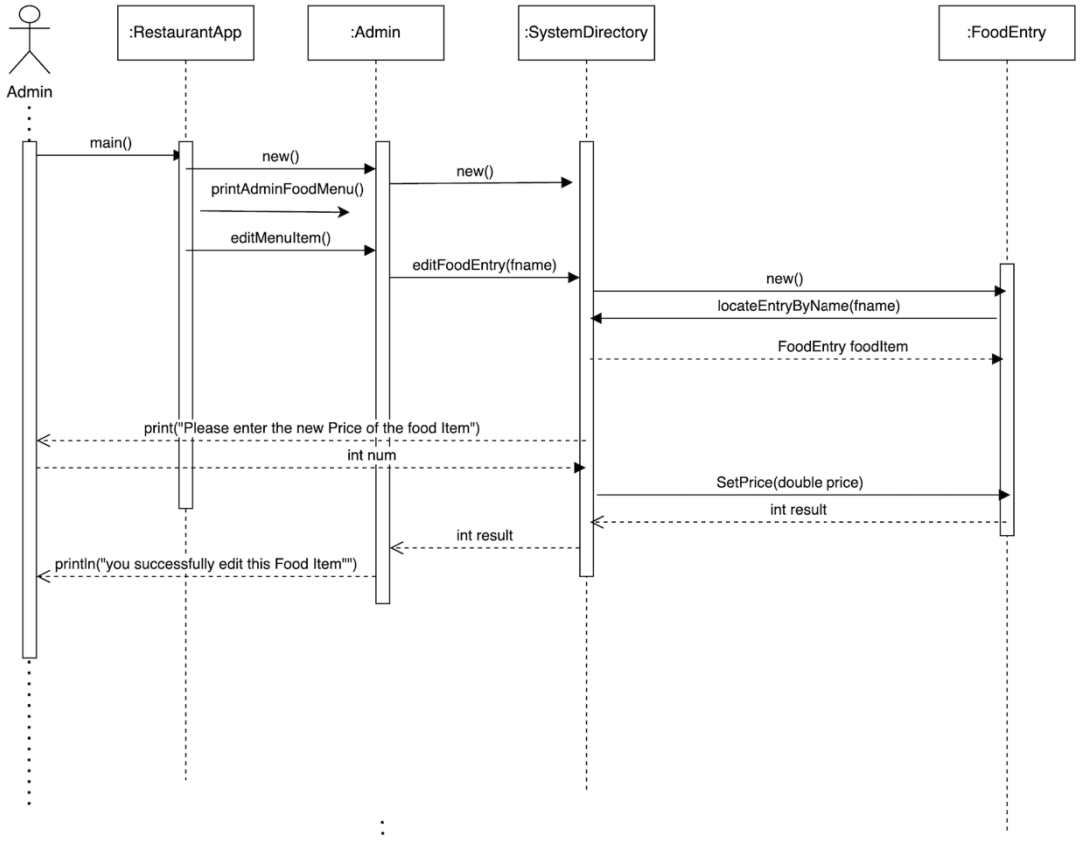
UC1 Add Menu item



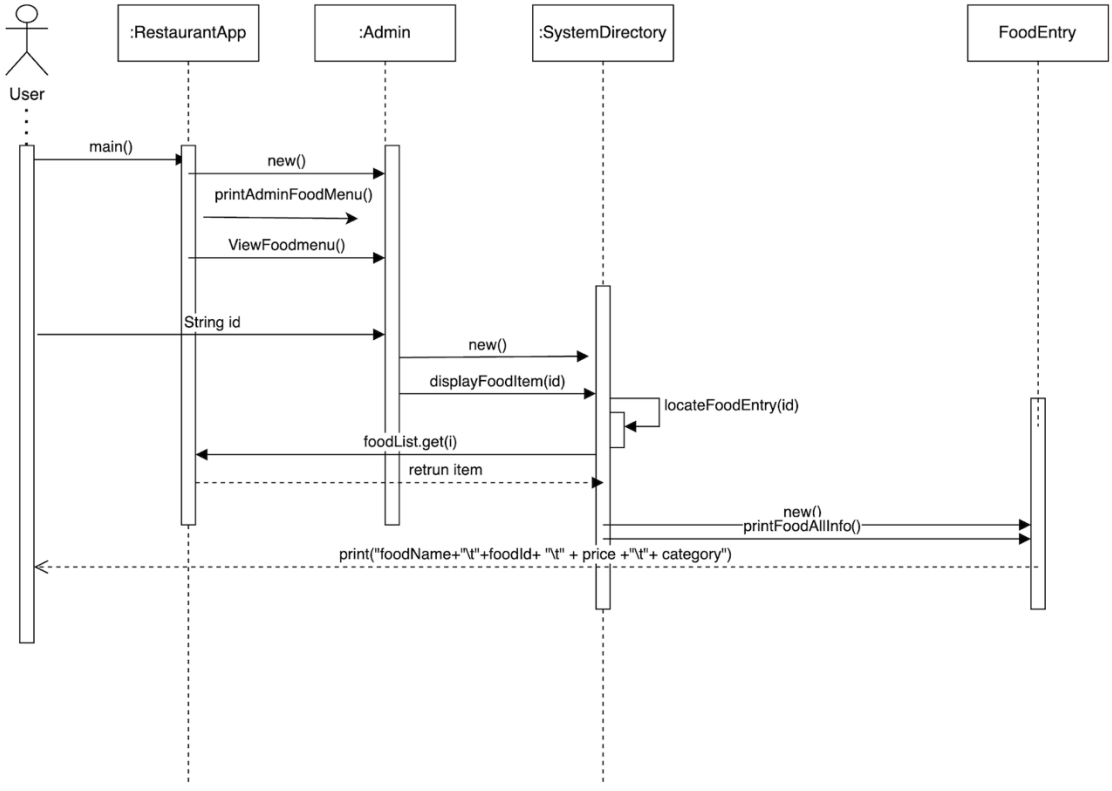
UC2 Delete Menu item



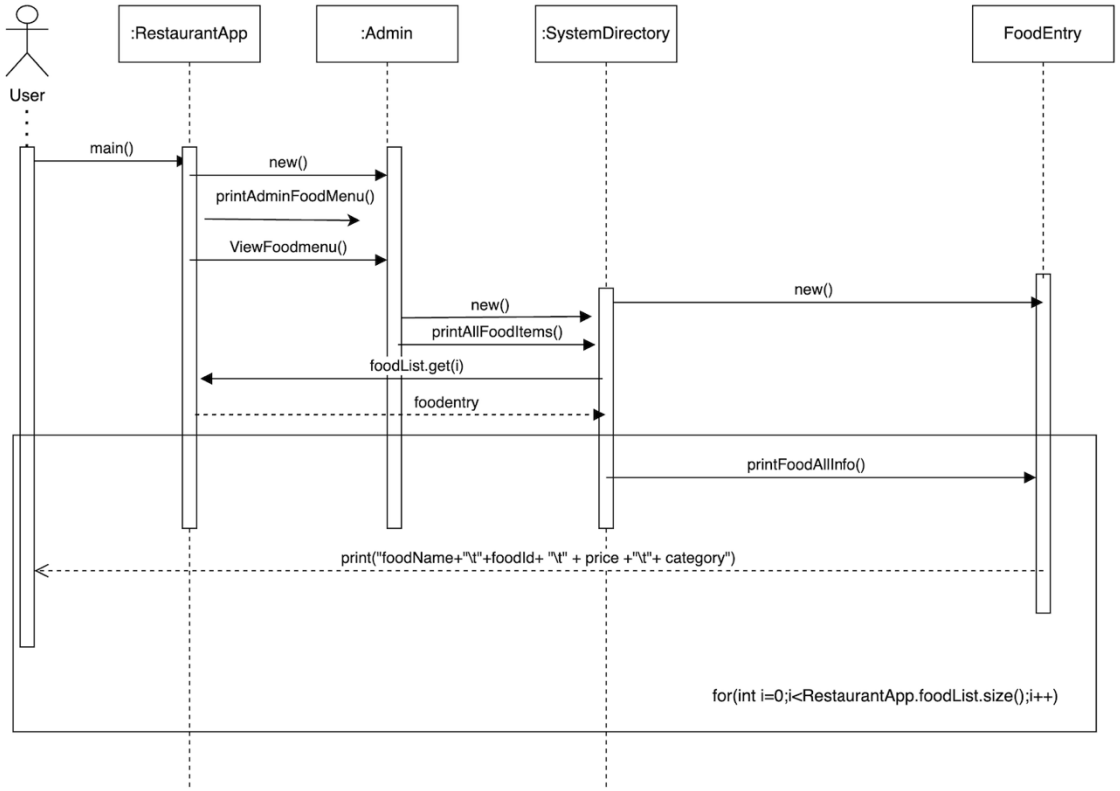
UC3 Edit a food item on the menu



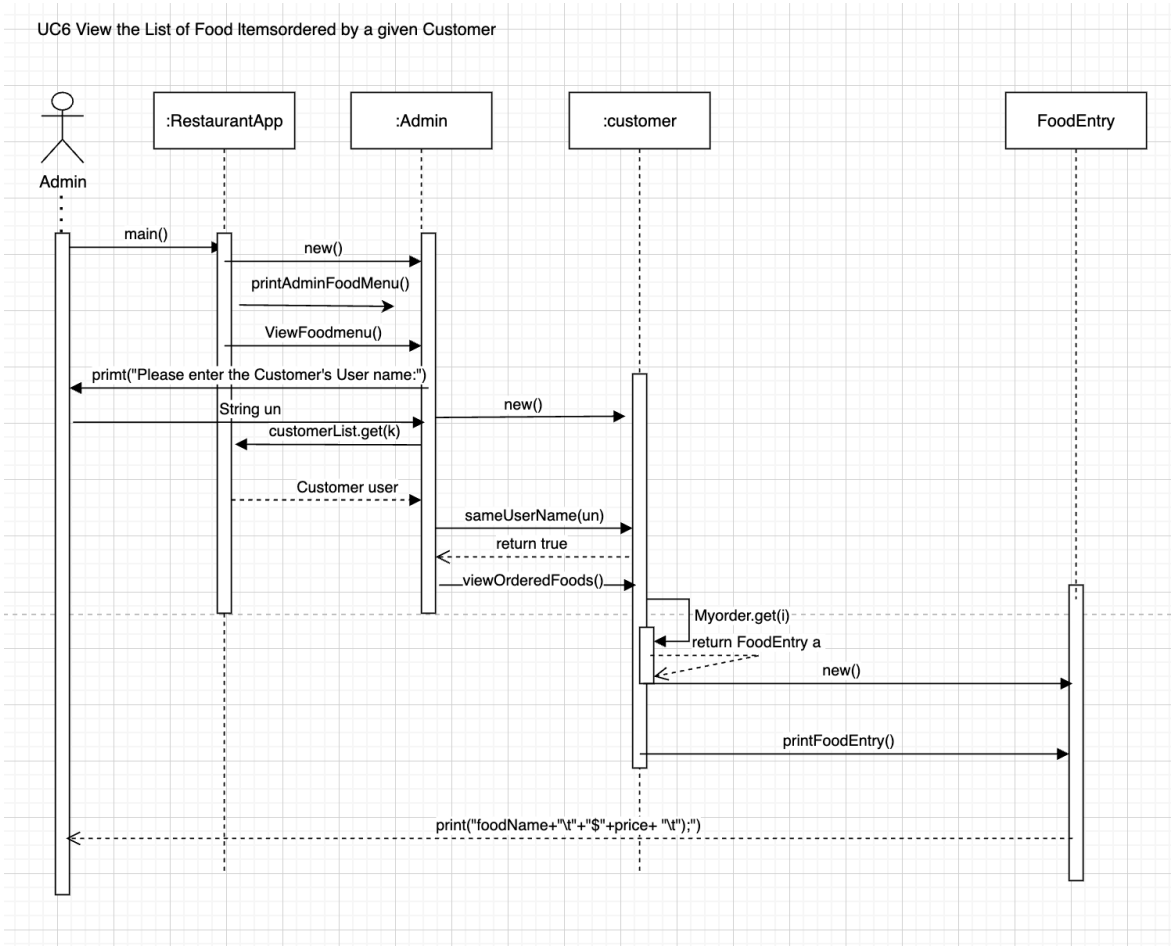
UC4 Display Information for a given Food Item



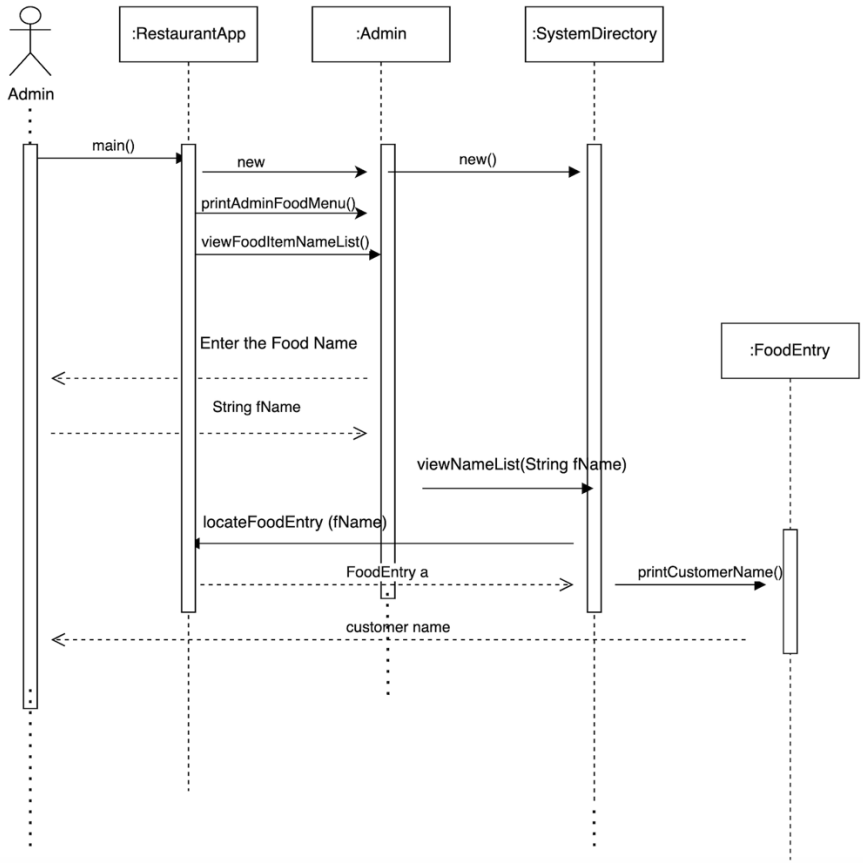
UC5 View all Food Items on theMenu



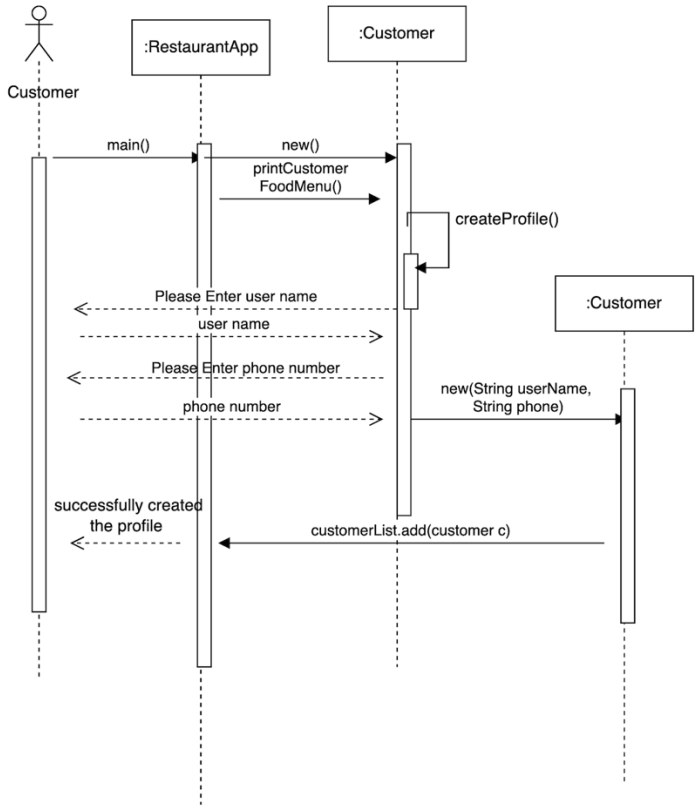
UC6 View the List of Food Itemsordered by a given Customer



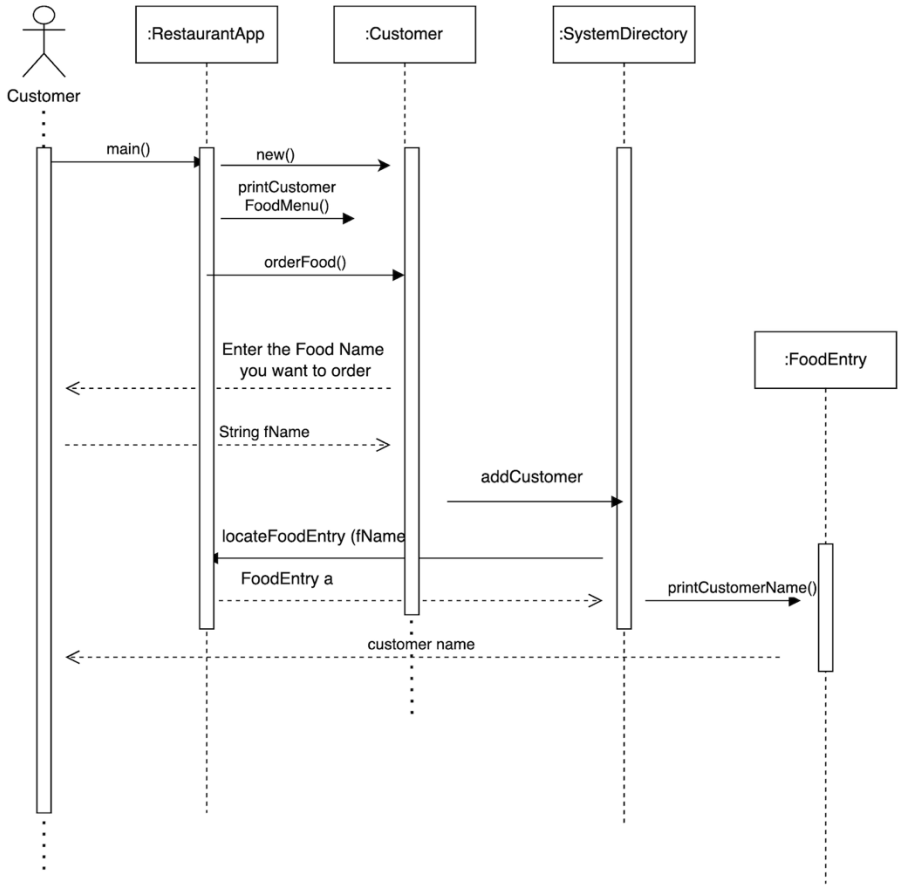
UC7 View the list of all customers that all ordered a given food



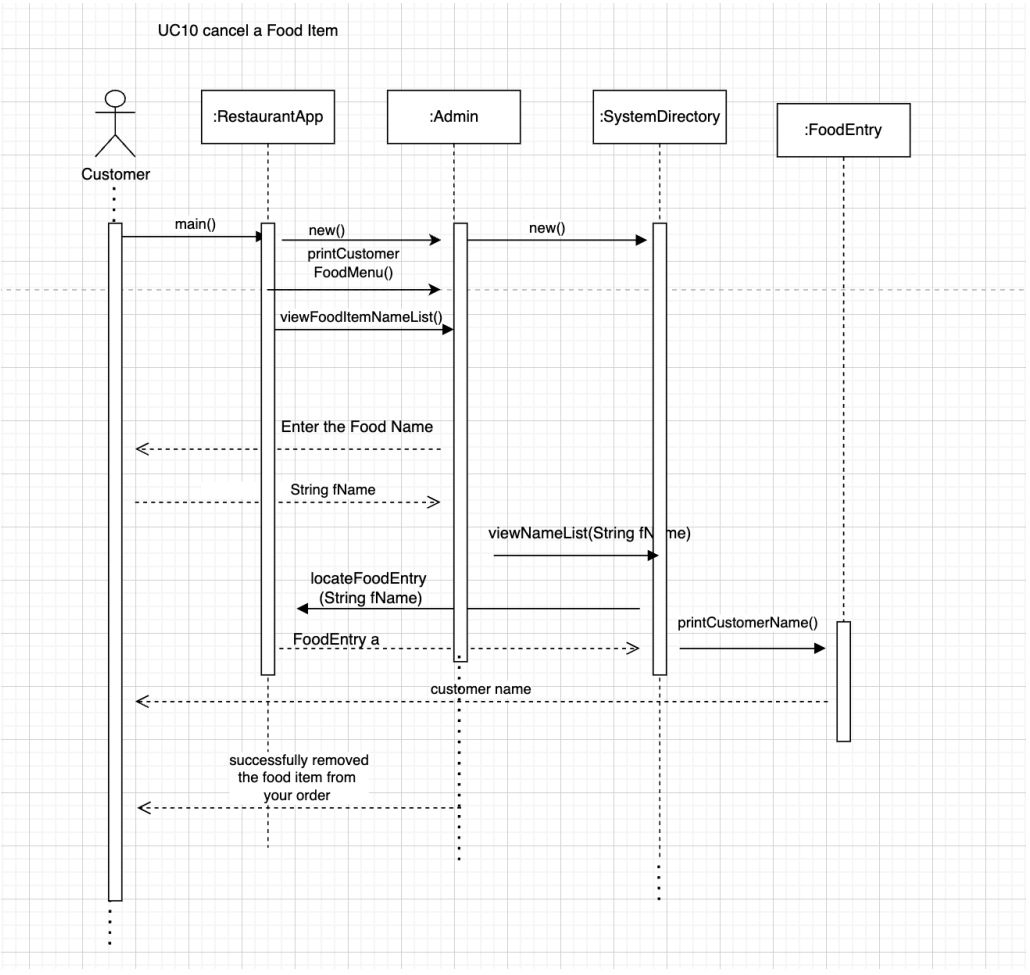
UC8 Create a user profile



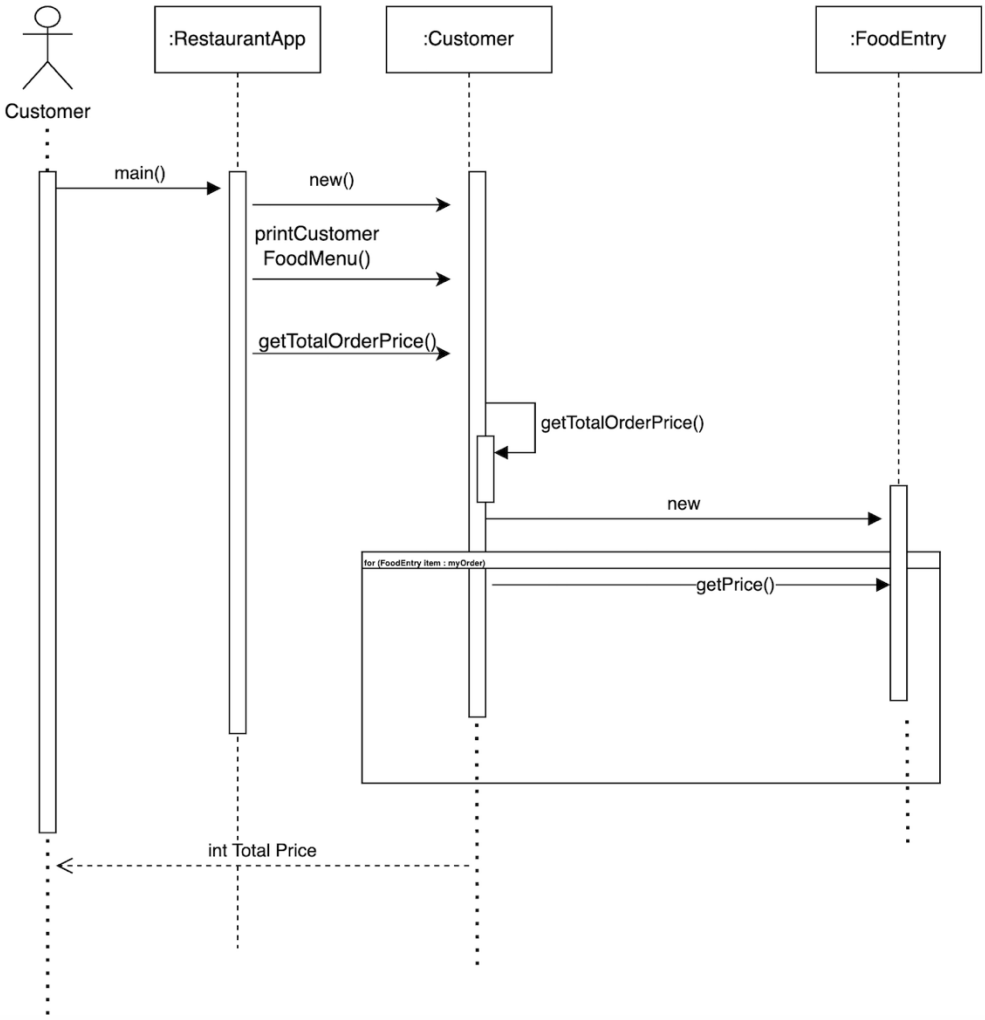
UC9 Order a Food item



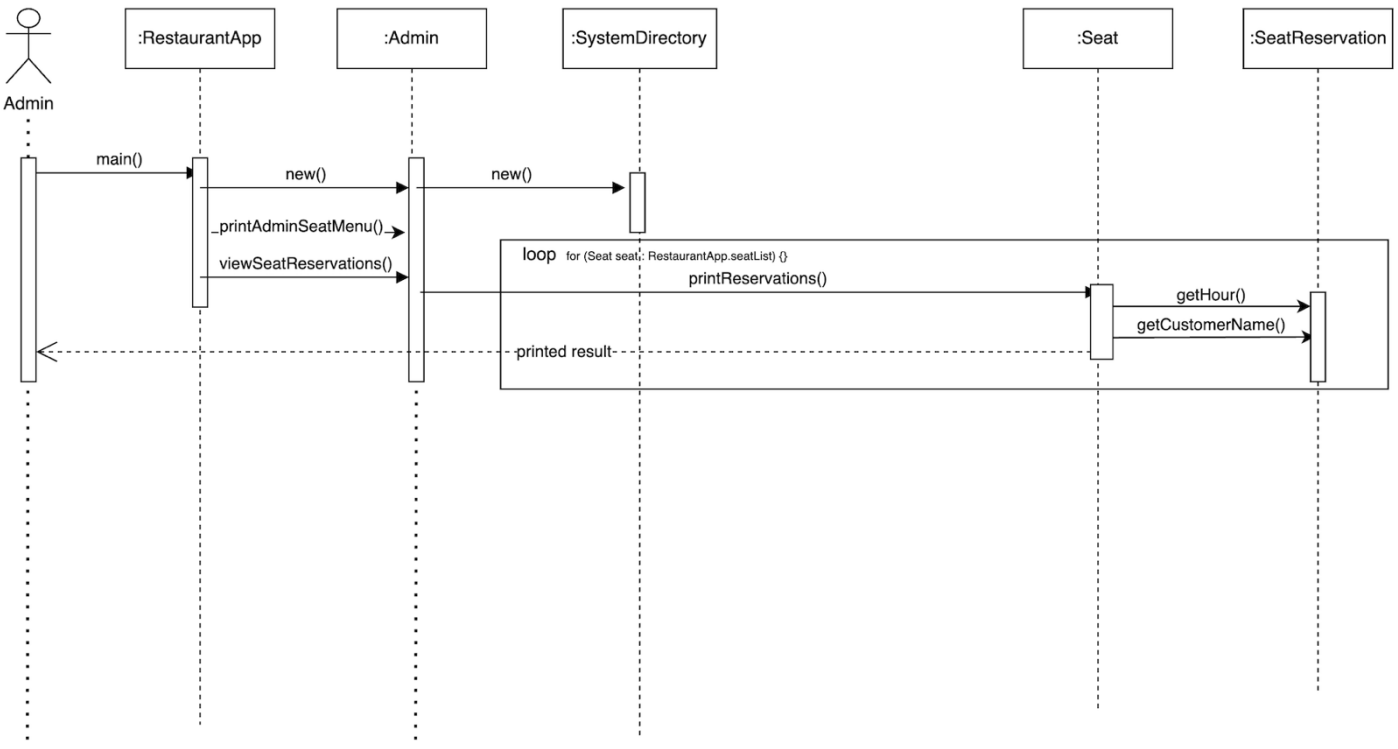
UC10 cancel a Food Item



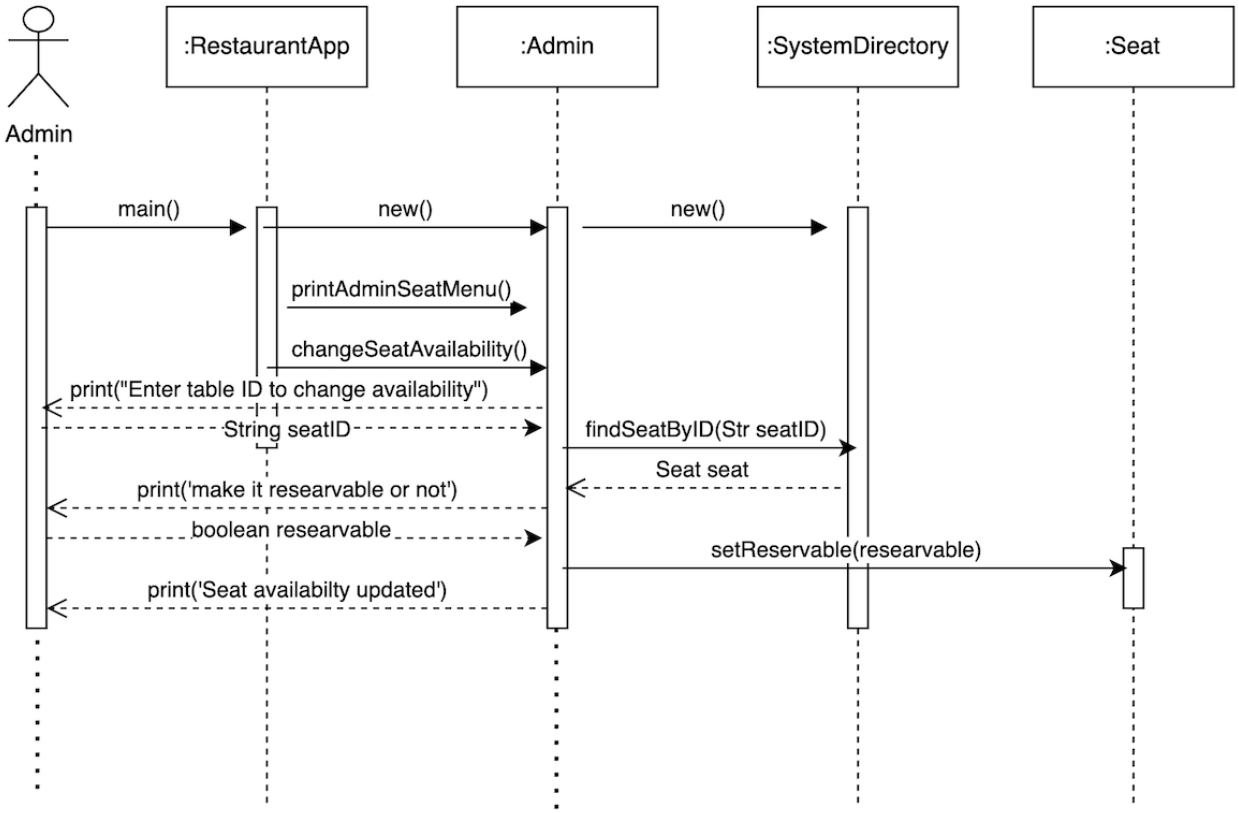
UC11 Calculate the total price of my order



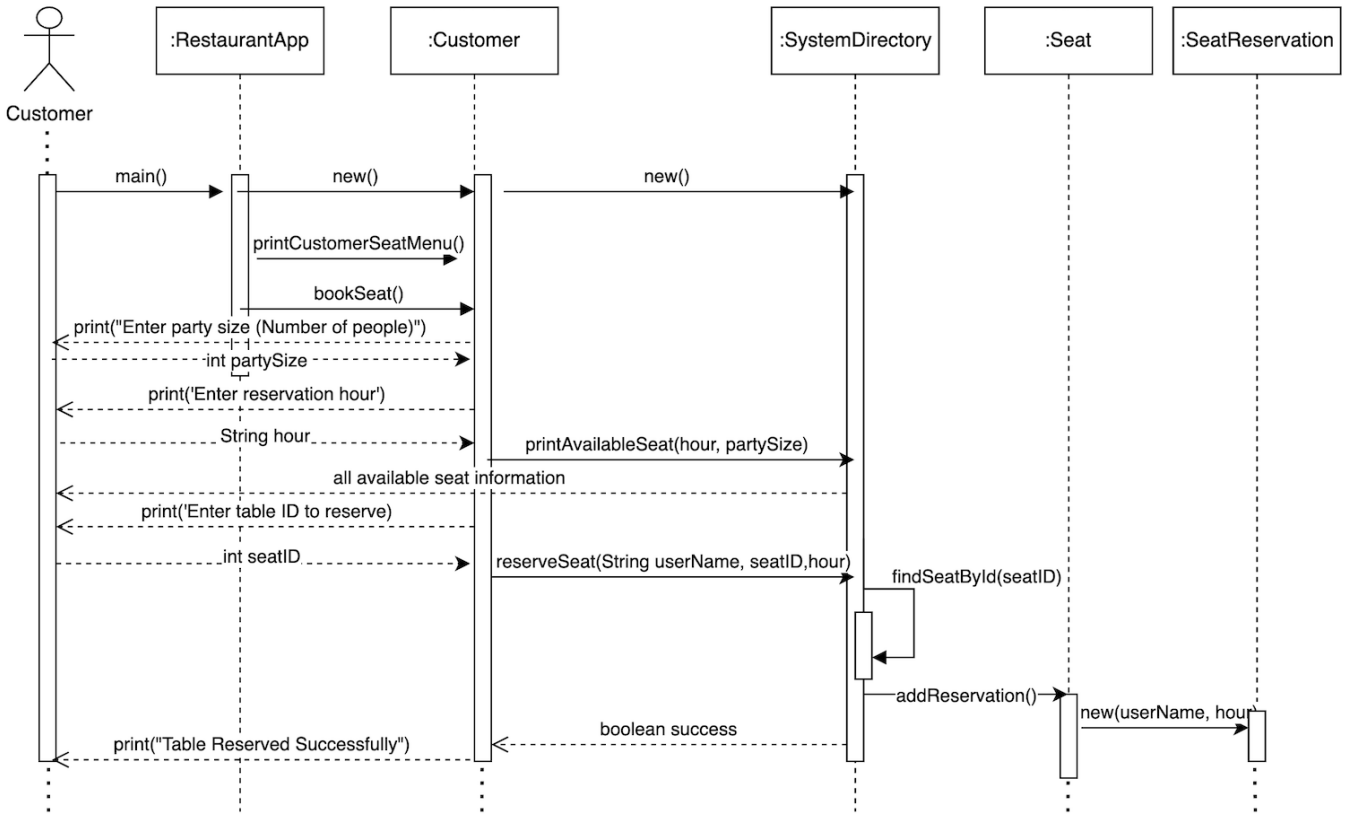
UC12 View All Table Reservation



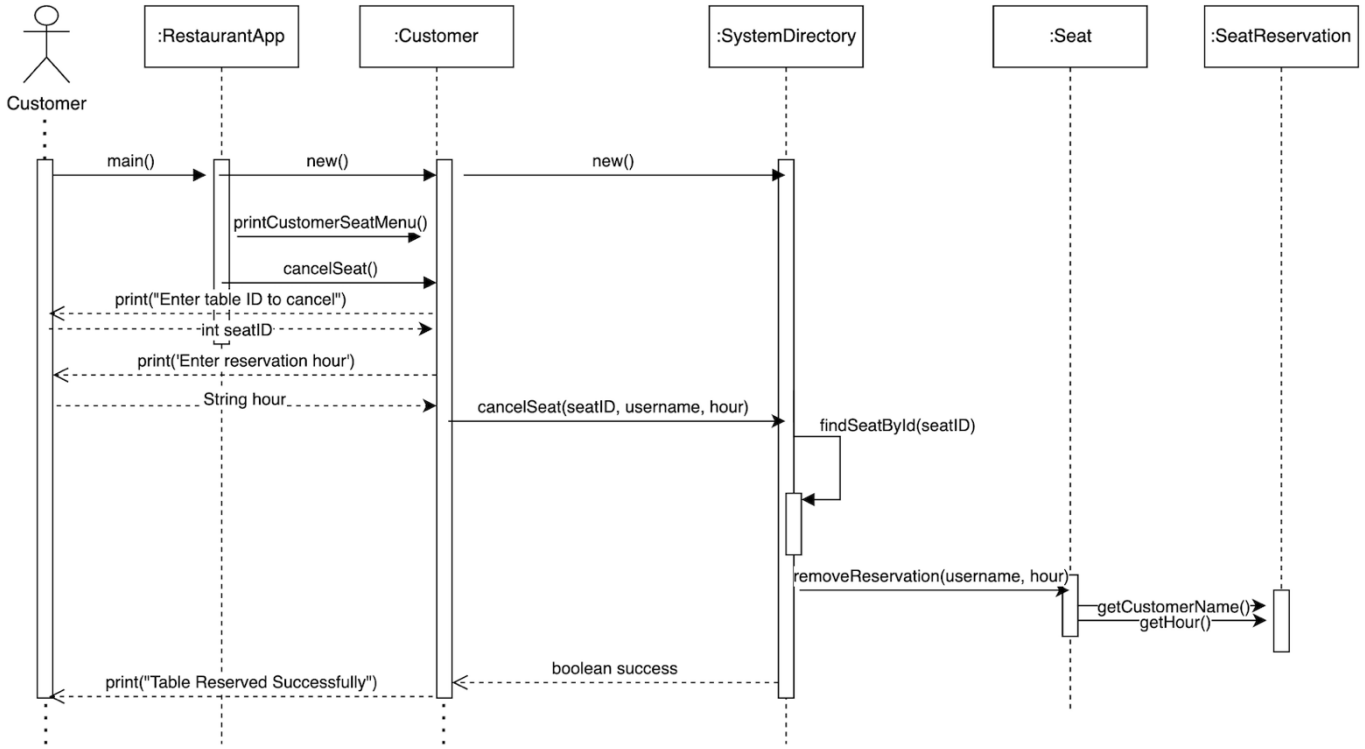
UC13 Change Table Availability



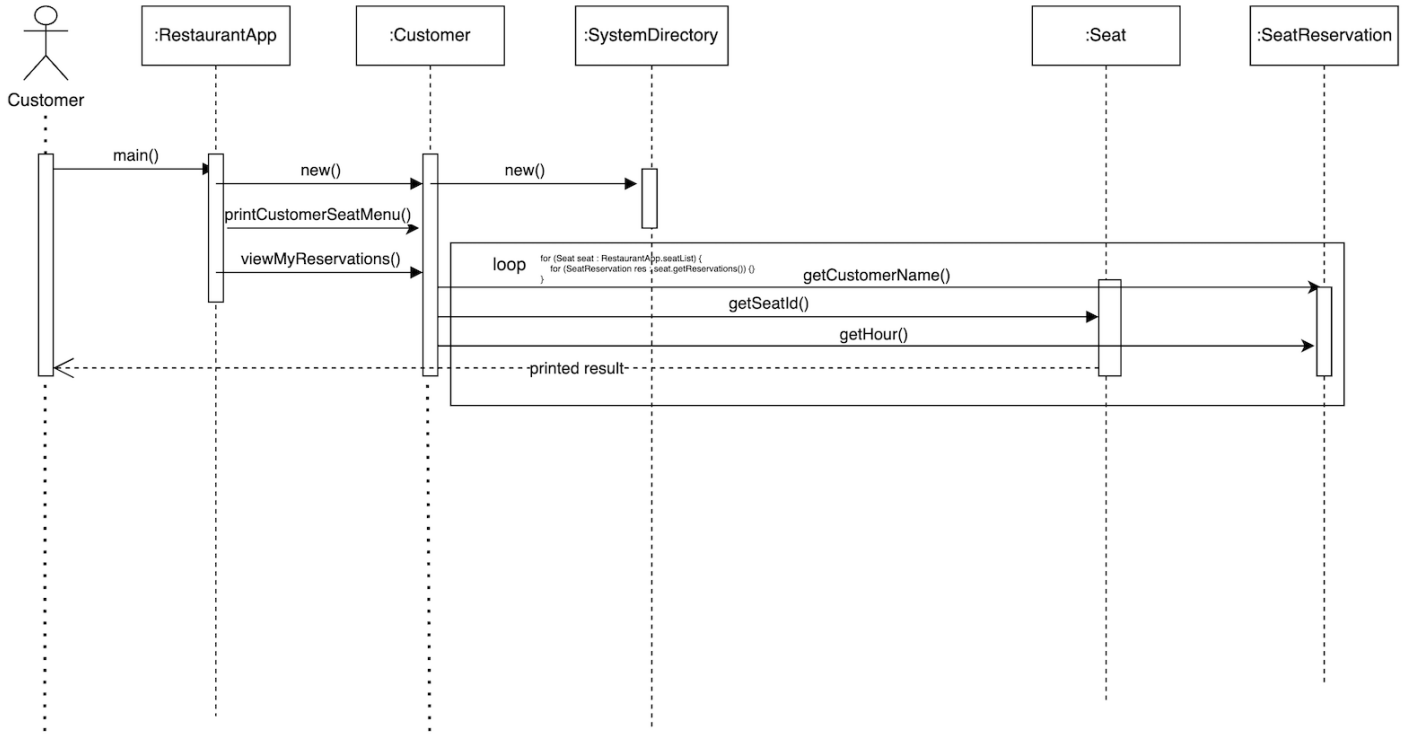
UC14 Book a Table



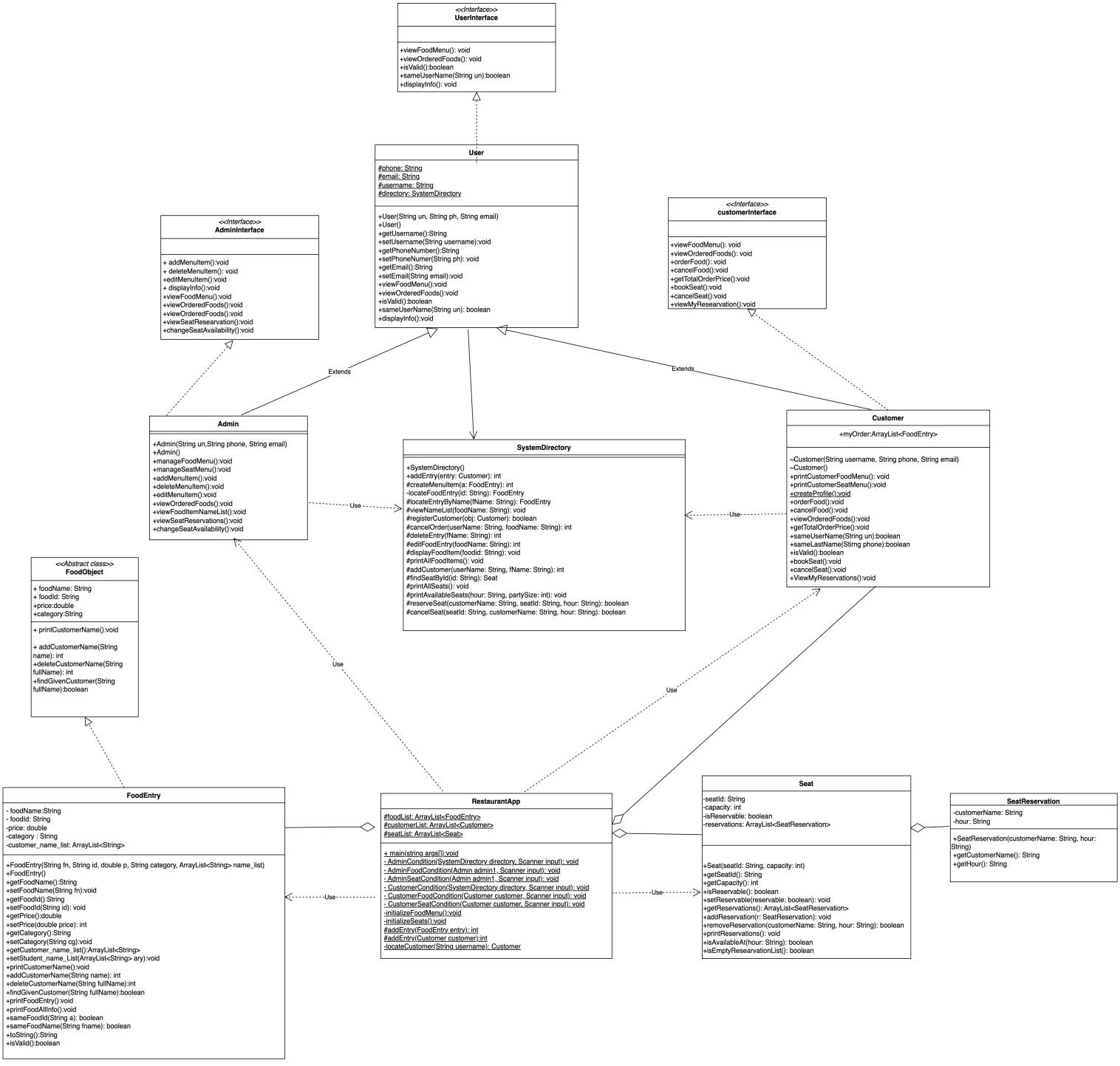
UC15 Cancel a Table



UC16 View my Table Reservation



Class Diagram



Conclusion:

Our Restaurant Food Ordering and Seat Booking System successfully demonstrates the practical application of OOP principles to solve a real-world problem. By clearly separating user roles and encapsulating responsibilities within well-structured classes, the system achieves both functional completeness and the code is good to maintain.

Although our currently design is based on a console-based, single-day-use application without long term storage, the architecture lays a strong foundation and our program is scalable for future development such as connecting with database to storage persistently, using graphical interfaces or web interface, and more complicated booking options like multi-day scheduling, and scheduling a table not only in integer point.

Appendix:

Our GitHub repository website is:

https://github.com/bangrui001/OOP_Final_Project_25Spring?tab=readme-ov-file#system-architecture