# Getting Ready: The Restaurant Management System

Understand the restaurant management system problem and learn the questions to simplify this problem further.

**We'll cover the following**

* [Expectations from the interviewee](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Expectations-from-the-interviewee)
  + [Restaurant services](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Restaurant-services)
  + [Restaurant management](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Restaurant-management)
* [Design approach](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-approach)
* [Design patterns](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-patterns)

A **restaurant management system** (RMS) is a software program that helps restaurant owners and managers to handle and organize various aspects of their businesses including food and beverage sales, employee schedules, and customer reservations. The restaurant management system can be used in various settings, including quick-service and fine-dining restaurants, and cafeterias. The system allows restaurants to streamline their operations, improve efficiency, and better serve their customers.

RMS is used to manage various aspects of a restaurant efficiently and securely. The restaurant’s management can have control and flexibility over their operations from a single location using this system. The system enables the manager to monitor available tables, make reservations, and generate bills.

Overall, the goal of a restaurant management system is to help restaurant owners and managers to streamline and automate various tasks, allowing them to focus on providing a high-quality dining experience to their customers.

An outer view of the restaurant

**1** of 3

## Expectations from the interviewee

There are several components in a restaurant management system, each with specific constraints and requirements. The following provides an overview of some of the main expectations that the interviewer will want to hear you discuss in more detail, during the interview.

### Restaurant services

To better understand the services offered by a restaurant management system, you may ask the interviewer the following questions:

1. Does the restaurant provide a delivery service?
2. Can a customer place an online order?
3. Does the restaurant accept online/card payments?

### Restaurant management

Restaurant branches vary in terms of services, which is why it is important to clear the following questions from the interviewer:

1. Can branches vary in terms of the services of a restaurant?
2. Do we need to consider the inventory management of the restaurant for this problem?

## Design approach

We’ll design this restaurant management system using the bottom-up approach. Therefore, we’ll follow the steps below:

* Identify and design the smallest components first—the table, table seat, meal, meal item, and seating chart.
* Use these small components to design bigger components—the menu, branch, and restaurant.
* Repeat the steps above until we design the whole restaurant management system.

## Design patterns

It is always a good practice to discuss the design patterns that the restaurant management system falls under, during the interview. Stating the design patterns will give the interviewer a positive impression and shows that the interviewee is well-versed in the advanced concepts of object-oriented design.

The following design patterns are used to design the restaurant management system:

* The Singleton design pattern
* The Factory design pattern

Let's explore the requirements of the restaurant management system in the next lesson.

# Requirements for the Restaurant Management System

Learn about all requirements of the restaurant management system.

**We'll cover the following**

* [Requirement collection](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Requirement-collection)

In this lesson, we’ll list the requirements of the restaurant management system design. This is a very crucial step, since requirements define the scope of a problem. Therefore, getting them right from the interviewer and understanding them well will make the design of the rest of the system smooth and easy.

We’ll use the notational convention to identify each requirement with a unique label, "Rn" where "R" is short for Requirement and "n" is a natural number.

## Requirement collection

**R1:** The restaurant can have multiple branches.

**R2:** Each branch will offer a menu with various sections and items.

**R3:** The waiter should be able to create an order for a table and add items for each person seated.

**R4:** Each person’s order can consist of multiple items, each corresponding to a menu item.

**R5:** The system should be able to provide information about tables currently available for walk-in customers.

**R6:** The system should allow for the reservation of tables.

**R7:** The receptionist should be able to search for available tables by date and time and make a reservation.

**R8:** The system should allow customers to make and cancel their reservations.

**R9:** The system should send notifications as the reservation time approaches.

**R10:** Customers should be able to pay their bills with credit cards, checks, or cash.

**R11:** Each branch may have different configurations of tables.

We've identified our requirements for the problem. In the next lesson, we’ll define different use cases of the restaurant management system.

# Use Case Diagram for the Restaurant Management System

Learn how to define use cases and create the corresponding use case diagram for the restaurant management system.

**We'll cover the following**

* [System](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#System)
* [Actors](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Actors)
  + [Primary actors](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Primary-actors)
* [Use cases](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Use-cases)
  + [Guest](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Guest)
  + [Manager](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Manager)
  + [Receptionist](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Receptionist)
  + [Waiter](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Waiter)
* [Relationships](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Relationships)
  + [Generalization](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Generalization)
  + [Associations](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Associations)
  + [Include](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Include)
* [Use case diagram](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Use-case-diagram)

Let’s build the use case diagram for the restaurant management system and understand the relationship between its different components.

First, we’ll define the different elements of our restaurant, followed by the complete use case diagram of the system.

## System

Our system is a "restaurant."

## Actors

Let’s define the main actors of our restaurant management system.

### Primary actors

* **Guest:** This is the restaurant's primary actor who can view the menu, place orders, and make payments.
* **Receptionist:** This actor is responsible for reserving tables and updating the table reservation status.
* **Waiter:** This actor takes the order from the guest and processes the payment.
* **Manager:** This actor acts as the admin of the system and can perform all tasks that a receptionist and a waiter can perform. Other than that, this actor updates the menu and sets the price of food items.

## Use cases

In this section, we’ll define the use cases for the restaurant management system. We’ve listed the use cases according to their respective interactions with a particular actor.

**Note:**You will see some use cases occurring multiple times, because they are shared among different actors in the system.

### Guest

* **View menu:** To view the food items available in the restaurant
* **Place order:** To place the food order in the restaurant
* **Add/update order item:** To add or remove a food item from the order
* **Cancel order:** To cancel an existing order
* **View order:** To view the item included in the order
* **Pay bill via cash:** To pay the food bill with cash
* **Pay bill via card:** To pay the food bill with the card
* **Pay bill via check:** To pay the food bill with the check
* **Reserve table:** To reserve a table for a guest
* **Update/cancel reservation:** To change the table or cancel the reservation

### Manager

* **Add/modify menu section:** To add a new section of the food type in the menu
* **Add/modify menu item:** To add a new food item in the food section
* **Set menu item price:** To set the price of a food item
* **Generate report:** To generate an analytical report of orders, inventory, and more
* **Add/ Update Tables chart:** To keep an updated record of the table availability status
* **Reserve table:** To reserve a table for a guest
* **Update/cancel reservation:** To change the table or cancel the reservation
* **View menu:** To view the food item available in the restaurant
* **Place order:** To place a food order in the restaurant
* **Add/update order item:** To add or remove a food item from the order
* **Cancel order:** To cancel an existing order
* **View order:** To view the item included in the order
* **Process payment:** To generate a food bill from the system and receive payment from the guest

### Receptionist

* **Add/ Update Tables chart:** To keep an updated record of the table availability status
* **Reserve table:** To reserve a table for a guest
* **Update/cancel reservation:** To change the table or cancel the reservation

### Waiter

* **View menu:** To view the food item available in the restaurant
* **Place order:** To place the food order in the restaurant
* **Add/update order item:** To add or remove a food item from the order
* **Cancel order:** To cancel an existing order
* **View order:** To view the item that is included in the order
* **Process payment:** To generate a food bill from the system and receive payment from the guest.

## Relationships

We describe the relationships between and among actors and their use cases in this section.

### Generalization

* The manager is responsible for the receptionist and the waiter. It also has access to everything they both have. Therefore, the “Manager” has a generalization relationship with both “Receptionist” and “Waiter.”

### Associations

The below table shows the association relationship between actors and their use cases.

|  |  |  |  |
| --- | --- | --- | --- |
| **Guest** | **Receptionist** | **Waiter** | **Manager** |
| Pay bill via card | Add/ Update Tables chart | View menu | Add/modify menu item |
| Pay bill via cash | Reserve table | View order | Add/modify menu section |
| Pay bill via check | Update/cancel reservation | Place order | Generate report |
| View menu |  | Add/update order item | Set menu item price |
| View order | Cancel order | View menu |
| Place order | Process Payment | View order |
| Add/update order item |  | Place order |
| Cancel order | Add/update order item |
| Reserve table | Cancel order |
| Update/cancel reservation | Add/ Update Tables chart |
|  | Reserve table |
| Update/cancel reservation |
| Print booking |
| Cancel booking |
| Process Payment |

### Include

* Whenever the manager adds a new menu item, the menu section is modified. Therefore, the “Add/ modify menu item” use case has an include relationship with the “add/modify menu section” use case.
* If the payment is processed, it will be either by card, cash, or check. Therefore, the “Process payment” use case has an extend relationship with the “Pay bill via cash,” “Pay bill via card,” and the “Pay bill via check” use cases.

## Use case diagram

Here’s the use case diagram of the restaurant management system:

The use case diagram of the restaurant management system.

In the next lesson, we’ll discuss the class diagram with a detailed explanation of all classes and their relationship with each other.

A diagram of a restaurant management system

Description automatically generated

# Class diagram for the Restaurant Management System

Learn to create a class diagram for the restaurant management system using the bottom-up approach.

**We'll cover the following**

* [Components of the restaurant management system](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Components-of-the-restaurant-management-system)
  + [Account](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Account)
  + [Person](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Person)
  + [Table and table seat](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Table-and-table-seat)
  + [Meal and meal item](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Meal-and-meal-item)
  + [Menu, menu section, and menu item](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Menu,-menu-section,-and-menu-item)
  + [Order](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Order)
  + [Kitchen](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Kitchen)
  + [Reservation](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Reservation)
  + [Payment](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Payment)
  + [Bill](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Bill)
  + [Notification](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Notification)
  + [Seating chart, branch, restaurant](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Seating-chart,-branch,-restaurant)
  + [Enumerations and custom data types](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Enumerations-and-custom-data-types)
* [Relationship between the classes](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Relationship-between-the-classes)
  + [Association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Association)
    - [One-way Association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#One-way-Association)
    - [Two-way Association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Two-way-Association)
  + [Composition](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Composition)
  + [Inheritance](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Inheritance)
* [Class diagram for the Restaurant Management System](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Class-diagram-for-the-Restaurant-Management-System)
* [Design pattern](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-pattern)
* [Additional requirements](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Additional-requirements)

We’ll create the class diagram for the restaurant management system. In the class diagram, we will first design the classes and then identify the relationship between classes according to the requirements for the restaurant management system design problem.

## Components of the restaurant management system

As mentioned earlier, we’ll follow the bottom-up approach to design a class diagram for the restaurant management system design.

### Account[**#**](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Account)

An Account contains an ID for the user. It also stores the address, account status, and password that can be reset.

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a menu

Description automatically generated

A screenshot of a menu

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### Association

The class diagram has the following association relationships:

#### One-way Association

* The Employee class has a one-way association with Branch.
* The Meal class has a one-way association with TableSeat.
* The MealItem class has a one-way association with MenuItem.
* The Waiter class has a one-way association with Order.
* The Order class has a one-way association with Table.
* The Chef class has a one-way association with Order and Kitchen.
* The Receptionist class has a one-way association with Reservation.
* The Reservation class has a one-way association with Table.

A screenshot of a diagram

Description automatically generated

A screenshot of a computer

Description automatically generated

A diagram of a company

Description automatically generated

* Both, Employee and Customer, extend the Person class, where the Employee class is extended by Chef, Manager, Waiter, and Receptionist.
* SMSNotification and EmailNotification extend the Notification class.
* Payment class is extended by CreditCardPayment, CashTransaction, and CheckTransaction.

**Note:** We have already discussed the inheritance relationship between classes in the component section above one by one.

**Class diagram for the Restaurant Management System**

A diagram of a company

Description automatically generated

## Design pattern

The following design patterns have been used in the class diagram:

* **The Singleton pattern:** This pattern ensures that a class has only one instance and provides a global point of access to that instance. This can be useful for the RMS’s database connection, for example, to ensure that there is only one connection to the database, and to make it easy for other parts of the system to access the connection.
* **The Command pattern:** This pattern encapsulates a request as an object, which allows the request to be parametrized with different data. In the context of an RMS, this pattern could be used to encapsulate requests to the database, such as add, delete or modify entries. This way the views or other components don’t need to know how to handle the request. They just need to know how to execute it.
* **The Observer pattern:** This pattern allows objects to be notified of changes to other objects. In the context of an RMS, this pattern could be used to notify the viewers of changes to the model so that they can update themselves accordingly.
* **The Factory pattern:** This pattern provides a way to create objects without specifying the exact class of object that will be created. In an RMS, this pattern could be used to create different types of menu items, for example, without having to specify the exact class of each item.

## Additional requirements

The interviewer can introduce some additional requirements in the given restaurant management system, or they can ask some follow-up questions. Let's see examples of the additional requirements:

**Discount:** A discount will be applied to the payment depending on special events such as the New Year, an anniversary, a branch opening, and so on. The class diagram provided below shows the relationship of Discount with the Payment class:

A screenshot of a computer screen

Description automatically generated

# Sequence Diagram for the Restaurant Management System

Create a sequence diagram for order modification in the restaurant management system and solve a challenge.

**We'll cover the following**

* [Modifying an order](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Modifying-an-order)
* [Sequence challenge: The guest pays for the order](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Sequence-challenge:-The-guest-pays-for-the-order)

Sequence diagrams are a great way to understand the interactions between different entities and objects in the system. There can be different sequence diagrams that we can create for our restaurant management system. In this lesson, we will create sequence diagrams for the following two interactions:

* Modifying an order
* **Sequence challenge:** The guest pays for the order.

## Modifying an order

The sequence diagram for modifying an order should have the following actors and objects that will interact with each other:

* **Actor:** Waiter
* **Objects:** Order, Kitchen, and bill

**Note:**Let us assume that the meal is not yet prepared.

Here’re the steps in the order modification interaction:

1. The waiter initiates the process by adding a meal item to the order.
2. The order is then updated. The updated order is sent to the kitchen for preparation.
3. The order is started to be prepared in the kitchen.
4. Simultaneously, the waiter updates the bill, as per the modified order.
5. After that, bill is updated.
6. The kitchen sends a message about the order indicating that it is prepared.
7. The order status is updated to "prepared."
8. The waiter serves the order.

Based on the order above, the sequence diagram of modifying an order in a restaurant management system is given below:

A screenshot of a computer screen

Description automatically generated

## Sequence challenge: The guest pays for the order

Let’s complete a sequence diagram for the order payment at the restaurant.

A skeleton of the sequence diagram, given that the payment is successful, is provided below:

A diagram of a product

Description automatically generated

Notice that the arrows in the diagram above are numbered from 1 to 5. The message boxes shown below are the messages to be exchanged between the actor(s) and object(s). Can you rearrange the messages below in the correct sequence they should appear in the skeleton of the sequence diagram above?

**Note:** If you are unsure, click the “Show Solution” button to check the correct answer.

Alternatively, click the "Show complete diagram" button to view the complete sequence diagram of the check-out interaction.

A diagram of a diagram

Description automatically generated

# Activity Diagram for the Restaurant Management System

Create some activity diagrams for the restaurant management system problem.

**We'll cover the following**

* [Place the order](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Place-the-order)
  + [States](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#States)
  + [Actions](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Actions)
* [Activity challenge: Cancel the order](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Activity-challenge:-Cancel-the-order)

Activities diagrams are a great way to visualize the flow of messages from one activity to the other in the system. There can be different activity diagrams that we can create for our restaurant management system. In this lesson, we’ll create activity diagrams for the following two activities:

* Place the order
* **Activity challenge:**How to cancel the order

## Place the order

The following are the states and actions that will be involved in this activity diagram.

### States

**Initial state:**A guest arrives at the restaurant and asks for a table.

**Final state:**The guest successfully placed a food order.

### Actions

The guest arrives at the restaurant reception and asks for a table. The receptionist checks for table availability. The guest will sit at the table if the table is available. If the table is unavailable, the guest will wait for a free table.

The waiter will appear with the menu when a guest sits at the table. The guest will decide to order from the menu. The waiter will send it to the kitchen.

Based on the order above, the activity diagram of the table booking and the food order are shown below:

A diagram of a restaurant

Description automatically generated

A diagram of a flowchart

Description automatically generated

The slots shown below represent the activities, and the arrows represent the flow from one activity to the other. Can you rearrange the slots below in the correct order they should appear in the activity diagram above?

A screenshot of a diagram

Description automatically generated

# Code for the Restaurant Management System

Write the object-oriented code to implement the design of the restaurant management system problem.

**We'll cover the following**

* [Restaurant management system classes](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Restaurant-management-system-classes)
  + [Enumerations and custom data type](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Enumerations-and-custom-data-type)
  + [Account and person](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Account-and-person)
  + [Table and table seat](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Table-and-table-seat)
  + [Meal and meal item](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Meal-and-meal-item)
  + [Menu, menu section, and menu item](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Menu,-menu-section,-and-menu-item)
  + [Order, kitchen, and reservation](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Order,-kitchen,-and-reservation)
  + [Payment and bill](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Payment-and-bill)
  + [Notification](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Notification)
  + [Seating chart, branch, and restaurant](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Seating-chart,-branch,-and-restaurant)
* [Wrapping up](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Wrapping-up)

We've covered different aspects of the restaurant management system and observed the attributes attached to the problem using various UML diagrams. Let us now explore the more practical side of things, where we will work on implementing the restaurant management system using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in the RMS:

* Java
* C#
* Python
* C++
* JavaScript

## Restaurant management system classes

In this section, we will provide the skeleton code of the classes designed in the class diagram lesson.

**Note:** For simplicity, we are not defining getter and setter functions. The reader can assume that all class attributes are private and accessed through their respective public getter methods and modified only through their public method functions.

### Enumerations and custom data type

The following code provides the definition of the enumeration and custom data type used in the restaurant management system.

PaymentStatus: This enumeration keeps track of an order's payment status by the customer.

TableStatus: This enumeration keeps track of the table’s status.

SeatType: This enumeration represents the type of seat for a customer.

AccountStatus: This enumeration keeps track of an account's status.

OrderStatus: This enumeration keeps the customer's order status in check.

ReservationStatus: This enumeration represents the reservation status of a table.

Address: This custom datatype represents the address of the restaurant's branch or a person.

**Note:** JavaScript does not support enumerations, so we will be using the Object.freeze() method as an alternative. This freezes an object and prevents further modifications.

Ja

// Enumerations

enum PaymentStatus {

Unpaid,

Pending,

Completed,

Failed,

Declined,

Canceled,

Abondoned,

Settling,

Refunded

}

enum TableStatus {

Free,

Reserved,

Occupied,

Other

}

enum SeatType {

Regular,

Kid,

Accessible,

Other

}

enum AccountStatus {

Active,

Closed,

Canceled,

Blacklisted

}

enum OrderStatus {

Received,

Preparing,

Complete,

Canceled,

None

}

enum ReservationStatus {

Requested,

Pending,

Confirmed,

CheckedIn,

Canceled,

Abondoned

}

// Custom Address data type class

public class Address {

private int zipCode;

private String address;

private String city;

private String state;

private String country;

}

### Account and person

The Account class represents a user account that has an ID, address, status, and a password that can be reset and is validated using the resetPassword() function.

The Person class stores the personal details of a person who can either be an Employee or a Customer of the restaurant. The derived class Employee is further extended by the following:

* Receptionist: Has a method named createReservation() to create reservations for customers
* Manager: Has a method named addEmployee() to hire new employees to the restaurant
* Chef: Has a method named prepareOrder() for preparing the order
* Waiter: Has a method named takeOrder() for taking new orders

The derived Customer class has a private member that stores the last visited date of the customer. The definitions of these classes are provided below:

Java

C#

Python

C++

JavaScript

public class Account {

private String accountId;

private String password;

private Address address;

private AccountStatus status;

public boolean resetPassword();

}

public abstract class Person {

private String name;

private String email;

private String phone;

}

public abstract class Employee extends Person {

private int employeeID;

private Date dateJoined;

private Account account;

}

public class Customer extends Person {

private Date lastVisitedDate;

}

public class Receptionist extends Employee {

public boolean createReservation();

}

public class Manager extends Employee {

public boolean addEmployee();

}

public class Chef extends Employee {

public boolean prepareOrder();

}

public class Waiter extends Employee {

public boolean takeOrder();

}

A screenshot of a computer program

Description automatically generated

A screenshot of a menu

Description automatically generated

MenuSection, where the MenuSection class is the representation of a menu's different sections. A MenuSectionhas a list of MenuItem in which an item's price can be updated. The definitions of these classes are provided below:

public class Menu {

private int menuID;

private String title;

private String description;

private double price;

private List<MenuSection> menuSections;

public boolean addMenuSection(MenuSection menuSection);

public boolean print();

}

public class MenuSection {

private int menuSectionID;

private String title;

private String description;

private List<MenuItem> menuItems;

public boolean addMenuItem(MenuItem menuItem);

}

public class MenuItem {

private int menuItemID;

private String title;

private String description;

private double price;

public boolean updatePrice(double price);

}

### Order, kitchen, and reservation

An Order has a number of meals and is assigned to a waiter and chef for a specific table. Meals can be updated in an Order using the addMeal() and removeMeal() methods.

A Kitchen has a number of chefs where new chefs can be assigned using the assignChef() method.

The Reservation class represents the reservation of a table that stores the reservation time, people count, status, check-in time, and customer information. It also maintains a list of notifications for the customer.

The definitions of these classes are provided below:

public class Order {

private int OrderID;

private OrderStatus status;

private Date creationTime;

private Meal[] meals;

private Table table;

private Waiter waiter;

private Chef chef;

public boolean addMeal(Meal meal);

public boolean removeMeal(Meal meal);

}

public class Kitchen {

private String name;

private Chef[] chefs;

public boolean assignChef();

}

public class Reservation {

private int reservationID;

private Date timeOfReservation;

private int peopleCount;

private ReservationStatus status;

private String notes;

private Date checkInTime;

private Customer customer;

private Table[] tables;

private List<Notification> notifications;

public boolean updatePeopleCount(int count);

}

### Payment and bill

The Payment class is an abstract class with the Check, CreditCard, and Cash classes as its child classes. This takes the PaymentStatus enum to keep track of the payment status. The Bill class represents the bill generated for a customer's order. The definitions of these classes are provided below:

Java

// Payment is an abstract class

public abstract class Payment {

private int paymentID;

private Date creationDate;

private double amount;

private PaymentStatus status;

public abstract void initiateTransaction();

}

public class Check extends Payment {

private String bankName;

private String checkNumber;

public void initiateTransaction() {

// functionality

}

}

public class CreditCard extends Payment {

private String nameOnCard;

private int zipcode;

public void initiateTransaction() {

// functionality

}

}

public class Cash extends Payment {

private double cashTendered;

public void initiateTransaction() {

// functionality

}

}

public class Bill {

private int billId;

private float amount;

private float tip;

private float tax;

private boolean isPaid;

public boolean generateBill();

}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Wrapping up**

We've explored the complete design of the restaurant management system in this chapter. We've looked at how a basic restaurant management system can be visualized using various UML diagrams and designed using object-oriented principles and design patterns.

Back