Restaurant Management System

A restaurant management is a software program that helps restaurant owners & managers to handle & organize various aspect of bussiness. RMS is used to manage restaurant & efficiently.

Expectations from the interviewer

→ Does the restaurant provide a delivery service?
→ Can a customer place an online order?
→ Does the restaurant accept online/card payment?
→ Can branches vary in terms of the services of a restaurant.
→ DO we ned to consider the inventory management of the restaurant for this problem?

Requirement collection RMS

R1: The restaurant can have multiple branches

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

R3: The waiter should be able to create an order for a table & add items for each person needed

R4: Each person's order can consists of multiple items, each corresponding to a menu items.

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

R6: The System should allow for the restaurant of tables

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

R8: The System should allow customers to make & cancel their reservation.

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 configuration of tables.

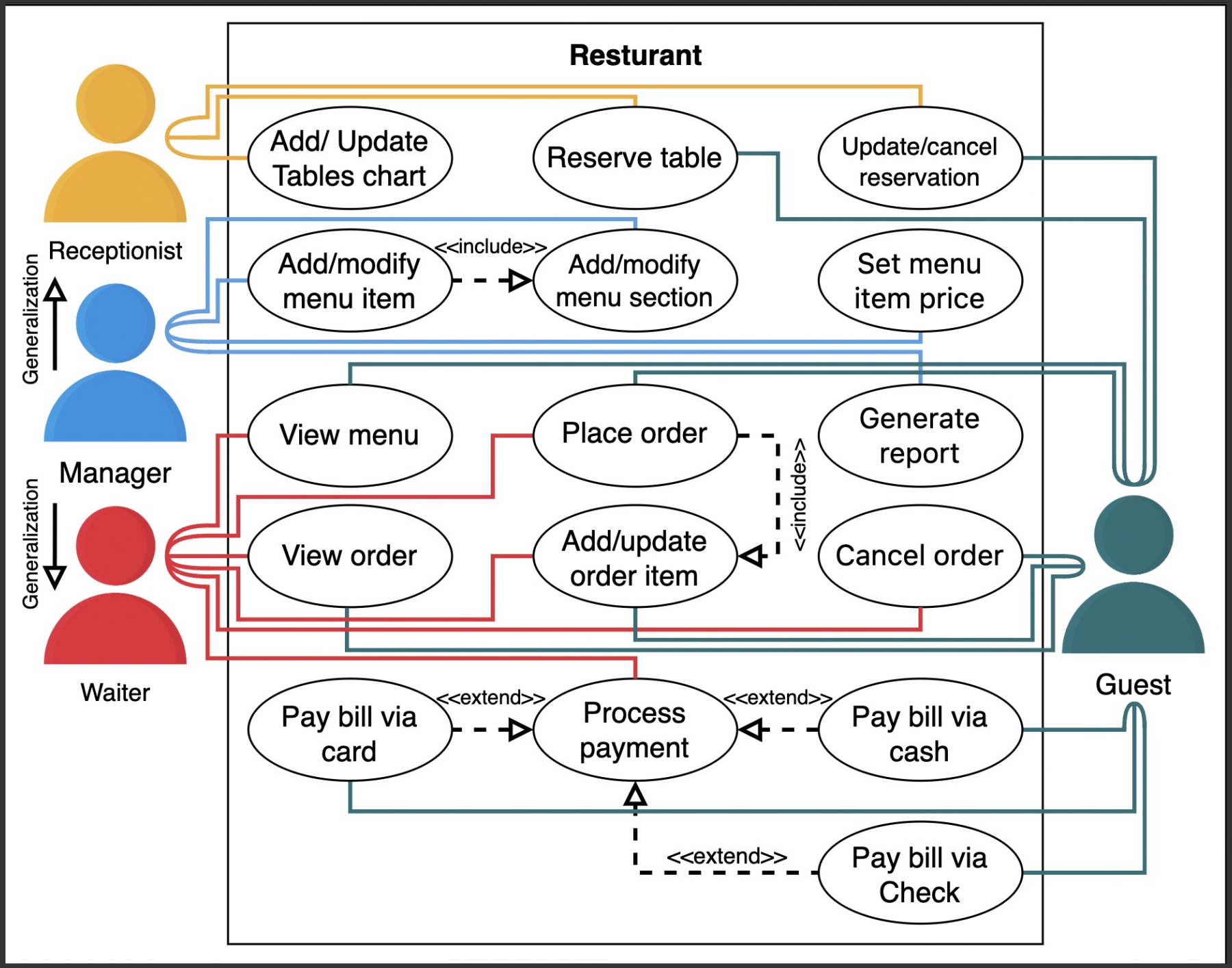
Actors

Primary Actors → Guest
→ Receptionist
→ waiter
→ manager

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

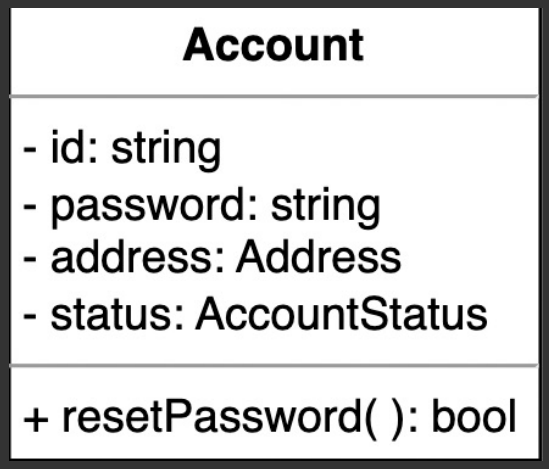
			Reserve table
			Update/cancel reservation
			Print booking
			Cancel booking
			Process Payment

Use Case Diagram

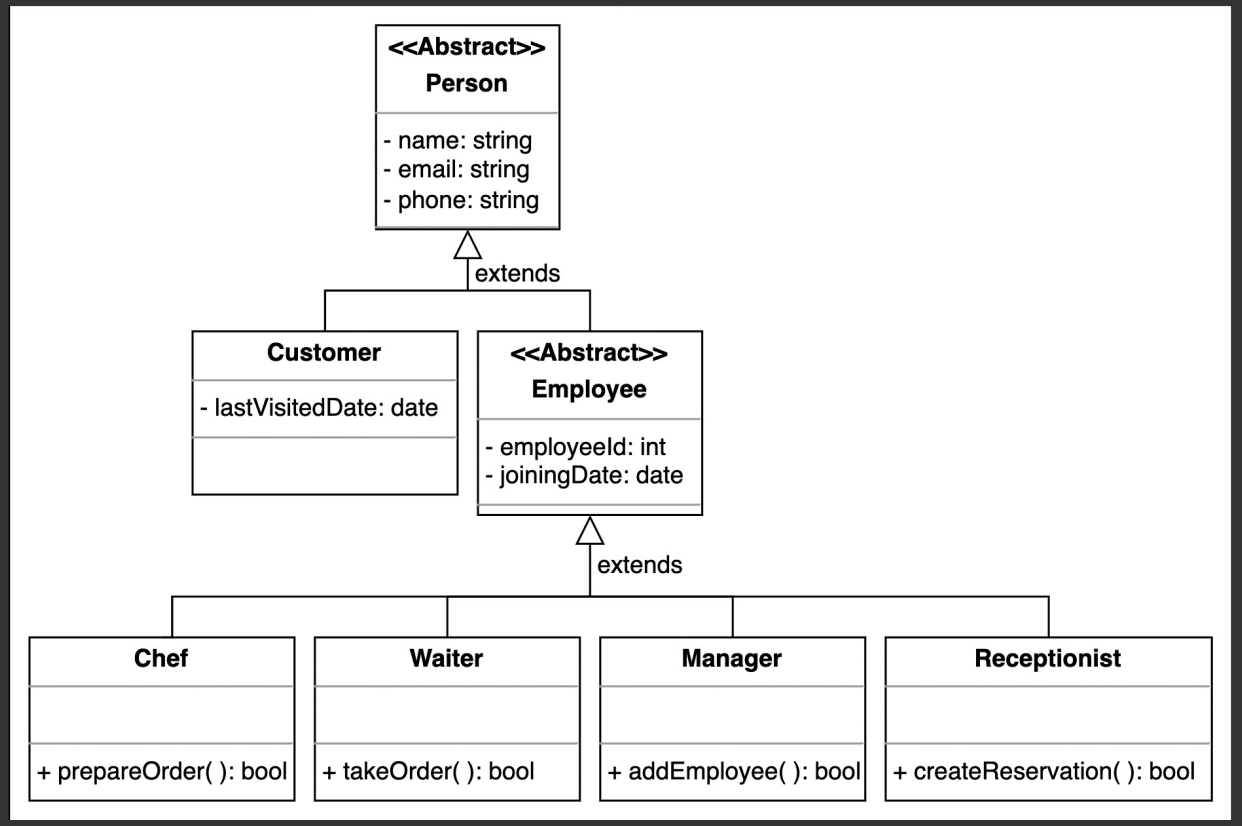


Class Diagram for RMS

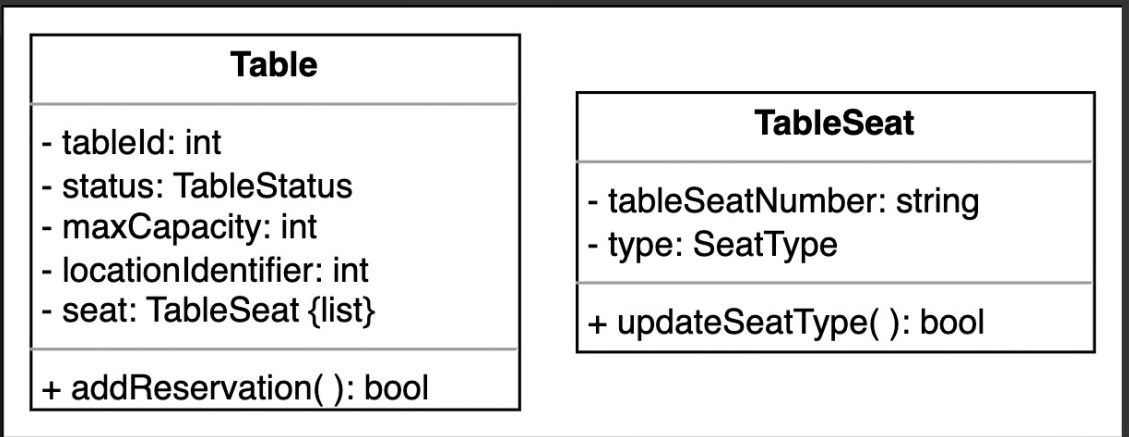
1. Account



2. Person



3. Table & Table Seat



4. meal & meal items

MealItem
- mealItemId: int - quantity: int
+ updateQuantity( ): bool

Meal
- mealId: int - mealItems: MealItem {list}
+ addMealItem( ): bool

5. Menu, menu section, & menu items

Menu
- menuId: int - title: string - description: string - price: double - menuSections: MenuSection {list}
+ addMenuSection( ): bool

MenuSection
- menuSectionId: int - title: string - description: string - menuItems: MenuItem {list}
+ addMenuItem( ): bool

MenuItem
- menuItemId: int - title: string - description: string - price: double
+ updatePrice( ): bool

6. Order

Order
- orderId: int - status: OrderStatus - meal: Meal {list} - creationDate: Date - table: Table - waiter: Waiter - chef: Chef
+ addMeal( ): bool + removeMeal( ): bool

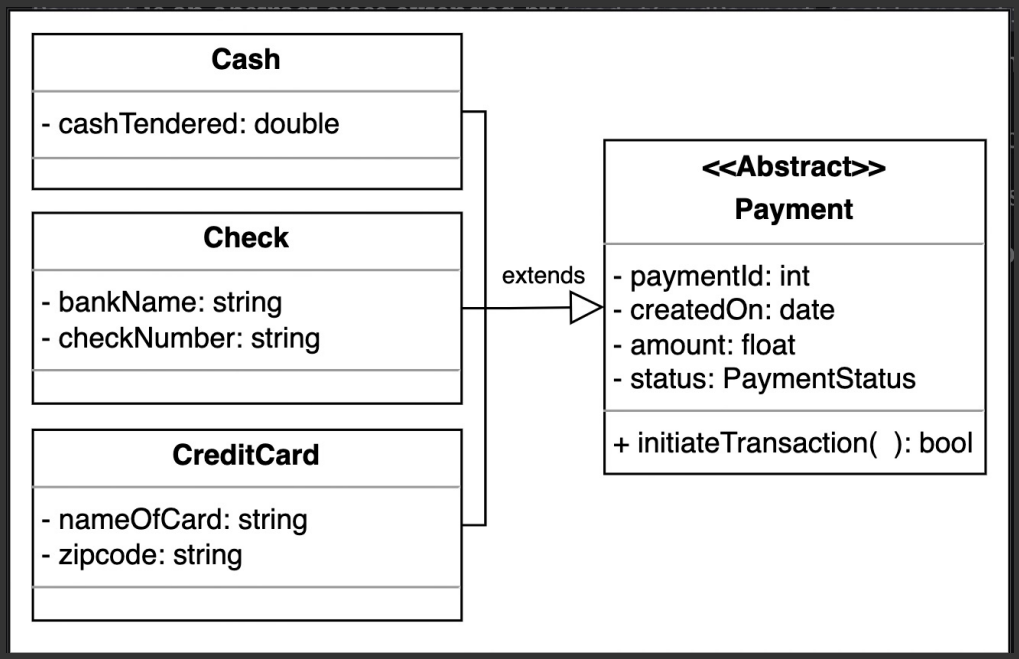
7. Kitchen

Kitchen
- name: string - chef: Chef {list}
+ assignChef( ): bool

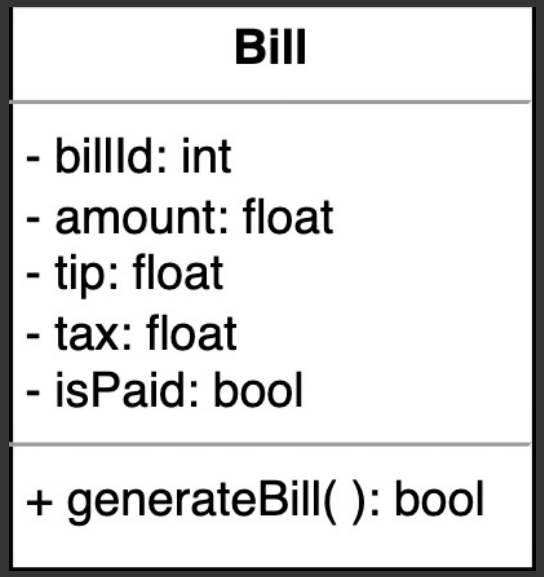
8. Reservation

Reservation
- reservationId: int - timeOfReservation: dateTime - peopleCount: int - status: ReservationStatus - notes: string - checkInTime: dateTime - customer: Customer
+ updatePeopleCount( ): bool

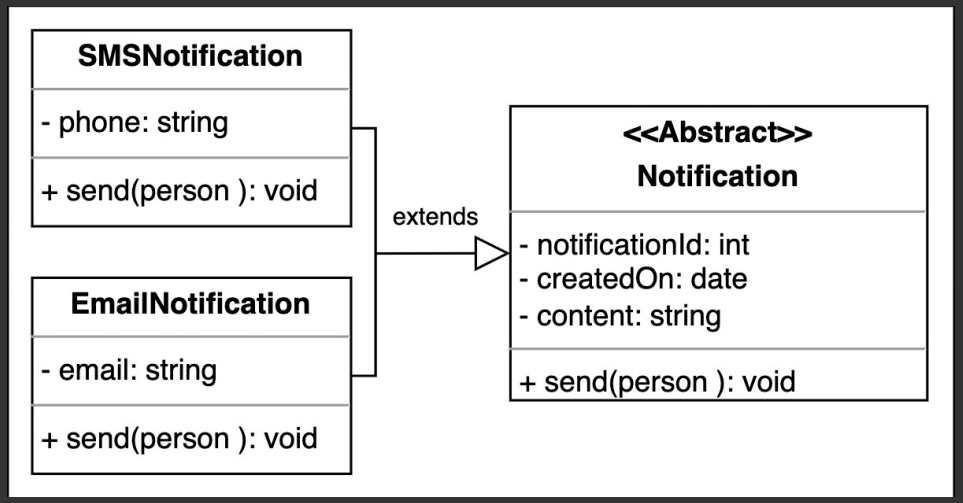
9. Payment



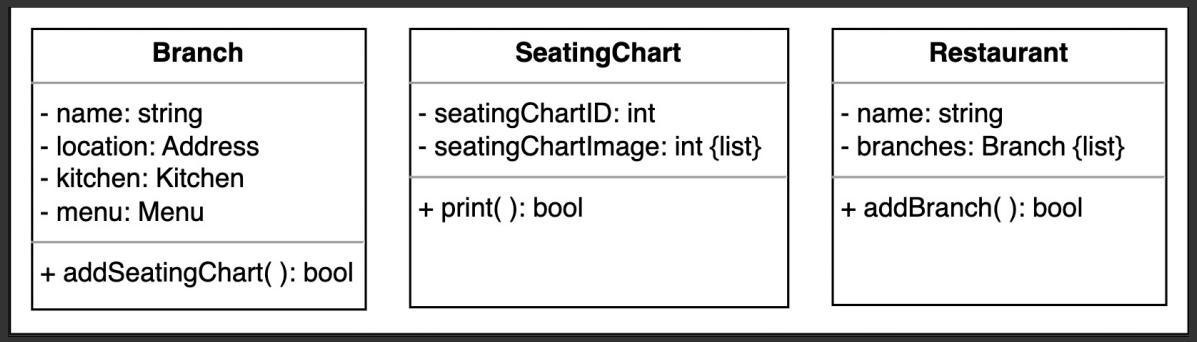
10. Bill



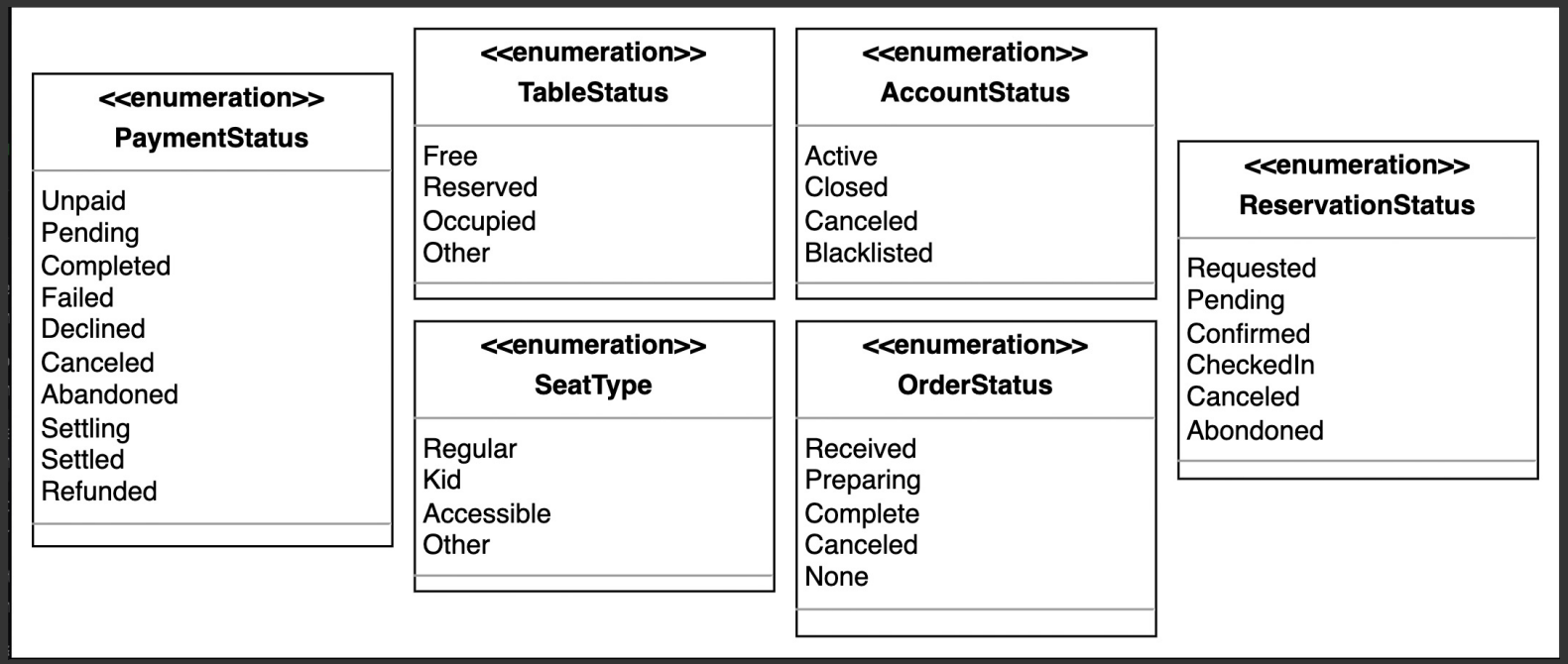
11. Notification



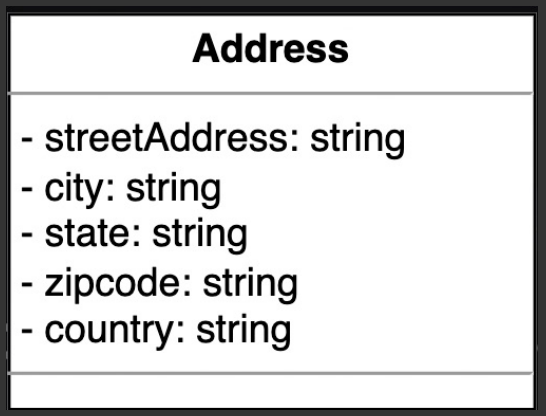
12. Seating chart, branch, restaurant



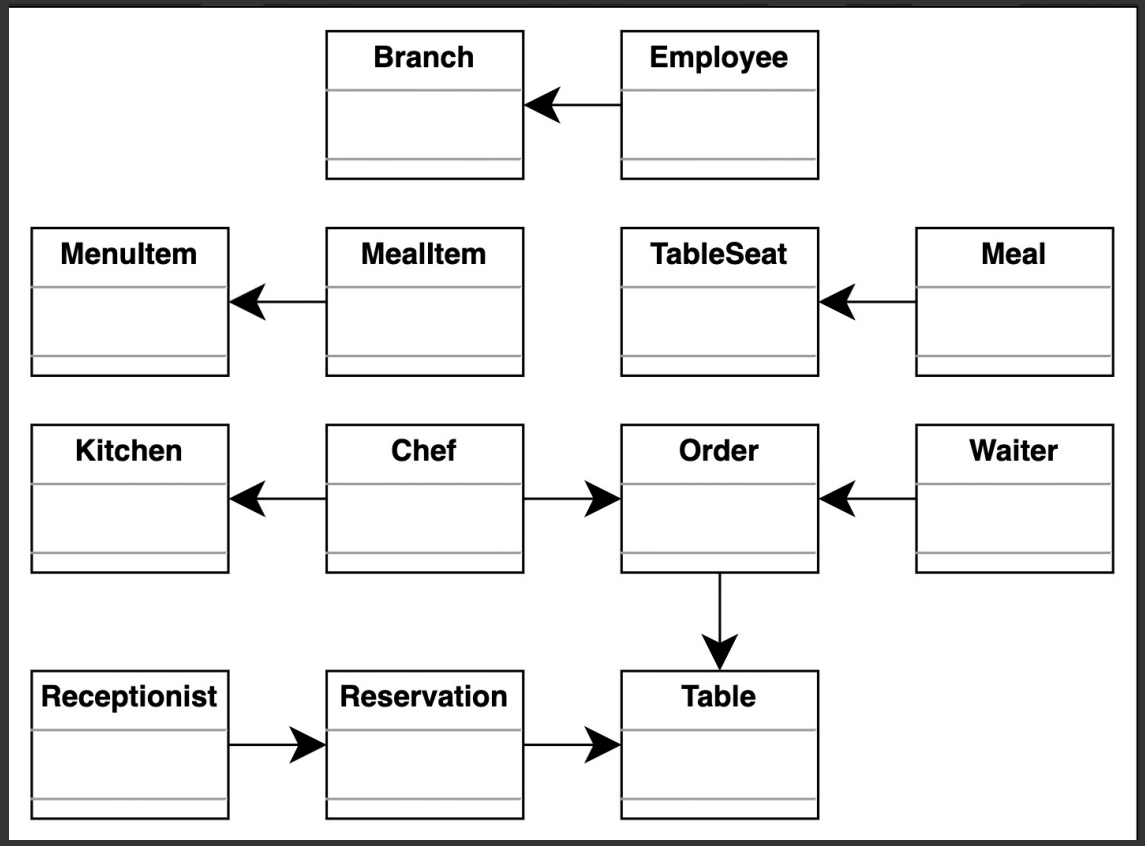
Enumerations



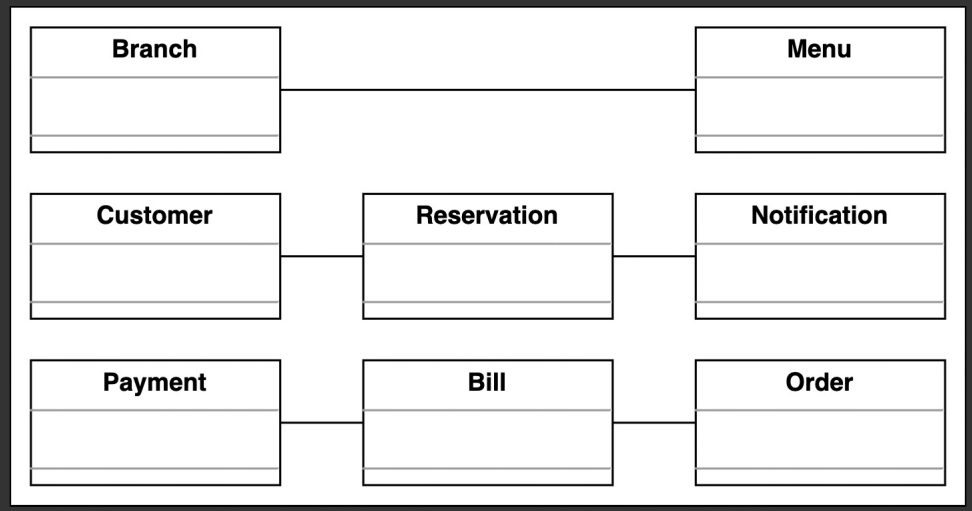
13. Address



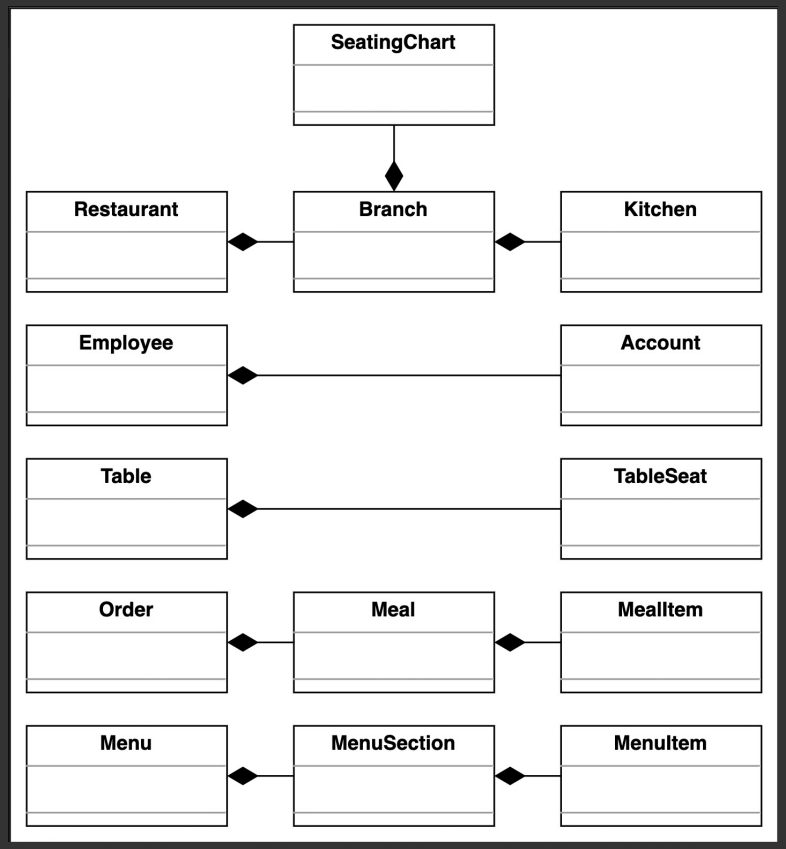
Associations

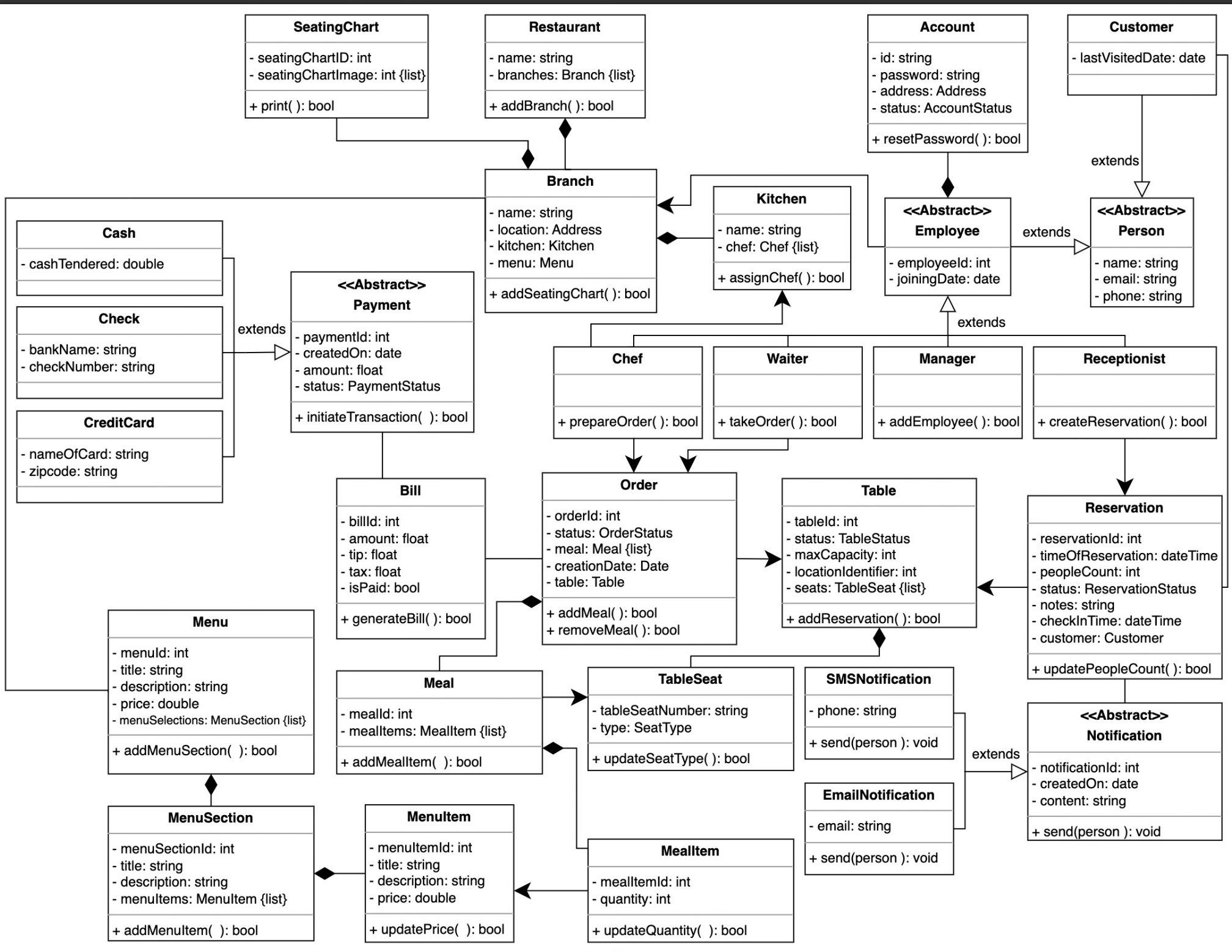


Two-way Association

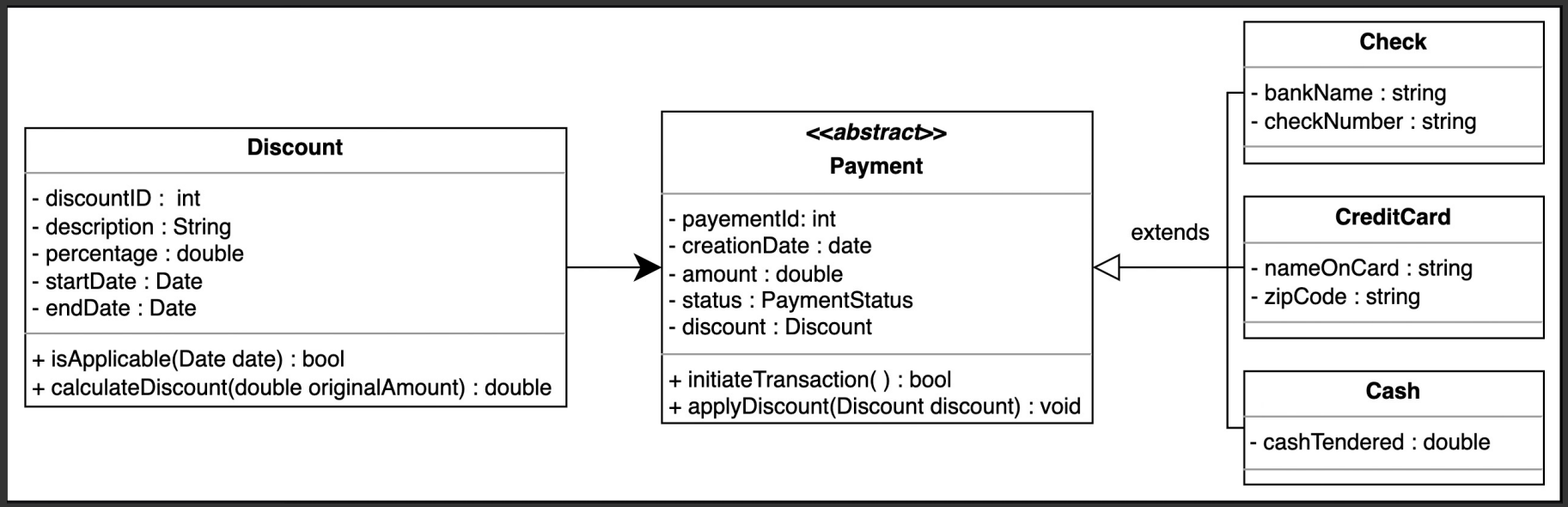


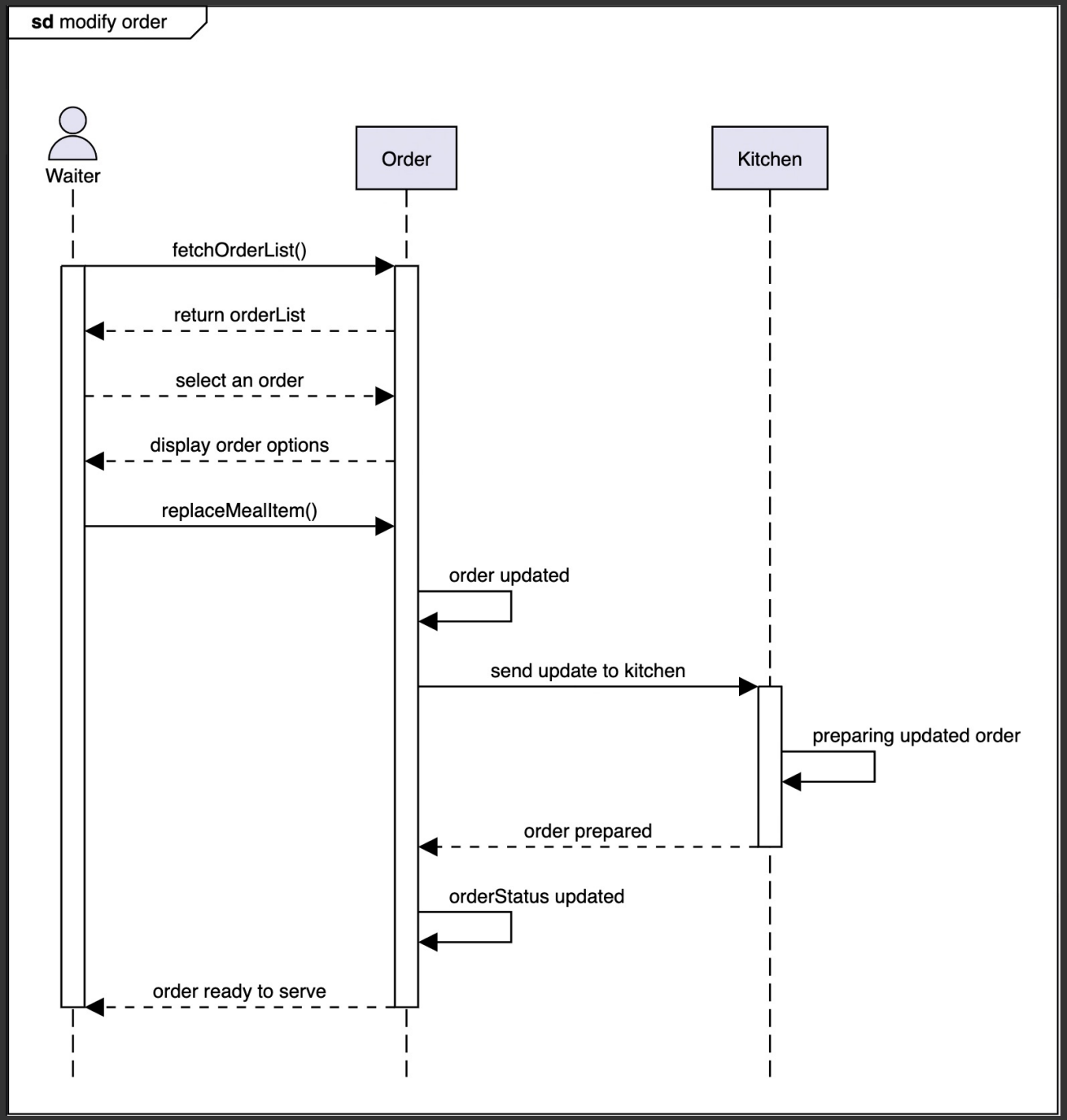
composición



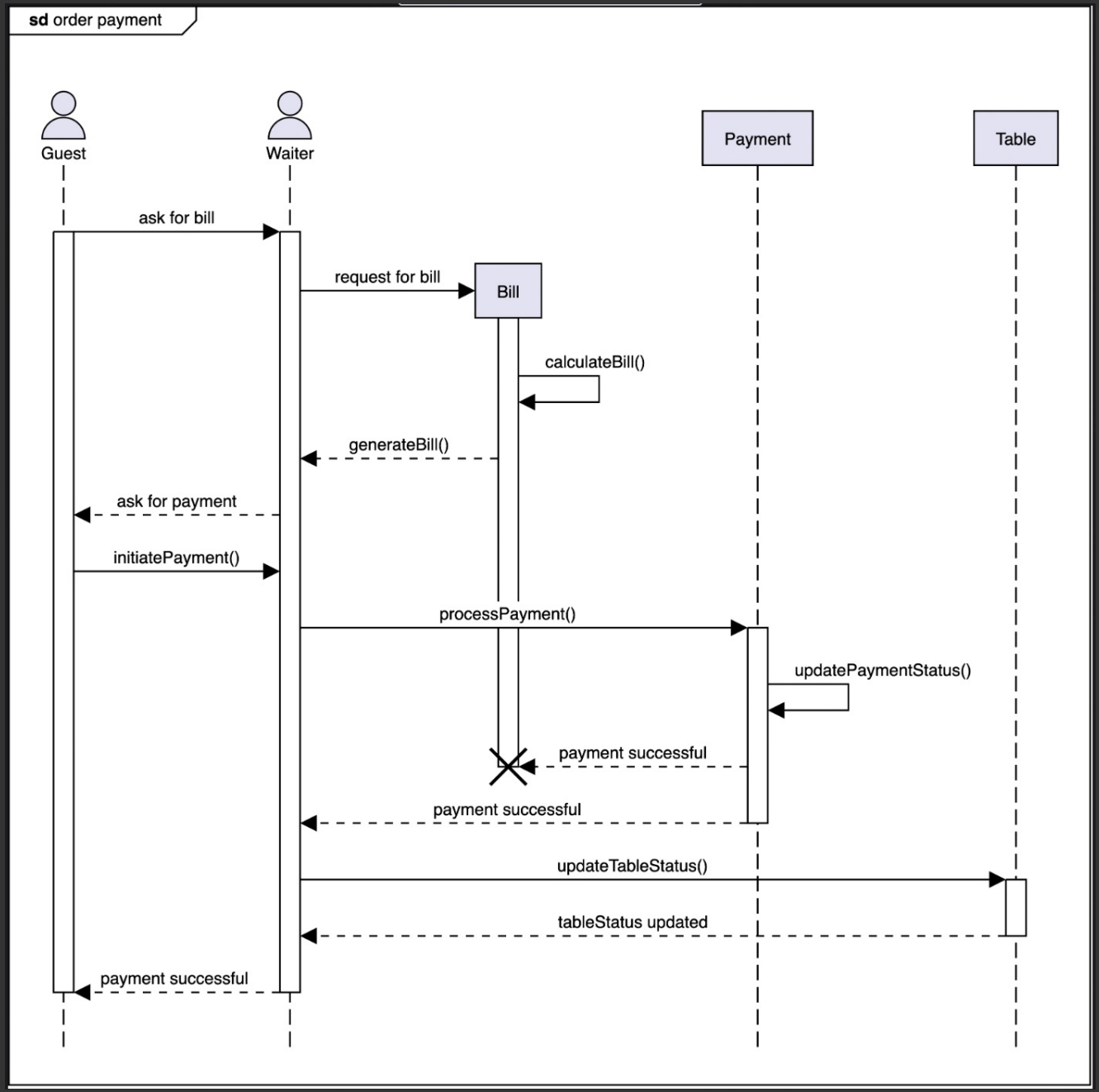

SeatingChart
- seatingChartID: int
- seatingChartImage: int {list}
+ print( ): bool
Restaurant
- name: string
- branches: Branch {list}
+ addBranch( ): bool
Account
- id: string
- password: string
- address: Address
- status: AccountStatus
+ resetPassword( ): bool
Customer
- lastVisitedDate: date
extends
Branch
- name: string
- location: Address
- kitchen: Kitchen
- menu: Menu
+ addSeatingChart( ): bool
Kitchen
- name: string
- chef: Chef {list}
+ assignChef( ): bool
<<Abstract>>
Employee
- employeeId: int
- joiningDate: date
extends
<<Abstract>>
Person
- name: string
- email: string
- phone: string
Cash
- cashTendered: double
Check
- bankName: string
- checkNumber: string
CreditCard
- nameOfCard: string
- zipcode: string
extends
<<Abstract>>
Payment
- paymentId: int
- createdOn: date
- amount: float
- status: PaymentStatus
+ initiateTransaction( ): bool
extends
Chef
+ prepareOrder( ): bool
Waiter
+ takeOrder( ): bool
Manager
+ addEmployee( ): bool
Receptionist
+ createReservation( ): bool
Bill
- billId: int
- amount: float
- tip: float
- tax: float
- isPaid: bool
+ generateBill( ): bool
Order
- orderId: int
- status: OrderStatus
- meal: Meal {list}
- creationDate: Date
- table: Table
+ addMeal( ): bool
+ removeMeal( ): bool
Table
- tableId: int
- status: TableStatus
- maxCapacity: int
- locationIdentifier: int
- seats: TableSeat {list}
+ addReservation( ): bool
Reservation
- reservationId: int
- timeOfReservation: dateTime
- peopleCount: int
- status: ReservationStatus
- notes: string
- checkInTime: dateTime
- customer: Customer
+ updatePeopleCount( ): bool
Menu
- menuId: int
- title: string
- description: string
- price: double
- menuSelections: MenuSection {list}
+ addMenuSection( ): bool
Meal
- mealId: int
- mealItems: MealItem {list}
+ addMealItem( ): bool
TableSeat
- tableSeatNumber: string
- type: SeatType
+ updateSeatType( ): bool
SMSNotification
- phone: string
+ send(person): void
EmailNotification
- email: string
+ send(person): void
extends
<<Abstract>>
Notification
- notificationId: int
- createdOn: date
- content: string
+ send(person): void
MenuSection
- menuSectionId: int
- title: string
- description: string
- menuItems: MenuItem {list}
+ addMenuItem( ): bool
MenuItem
- menuItemId: int
- title: string
- description: string
- price: double
+ updatePrice( ): bool
MealItem
- mealItemId: int
- quantity: int
+ updateQuantity( ): bool

Addinal Requirement ⇒ Discount.

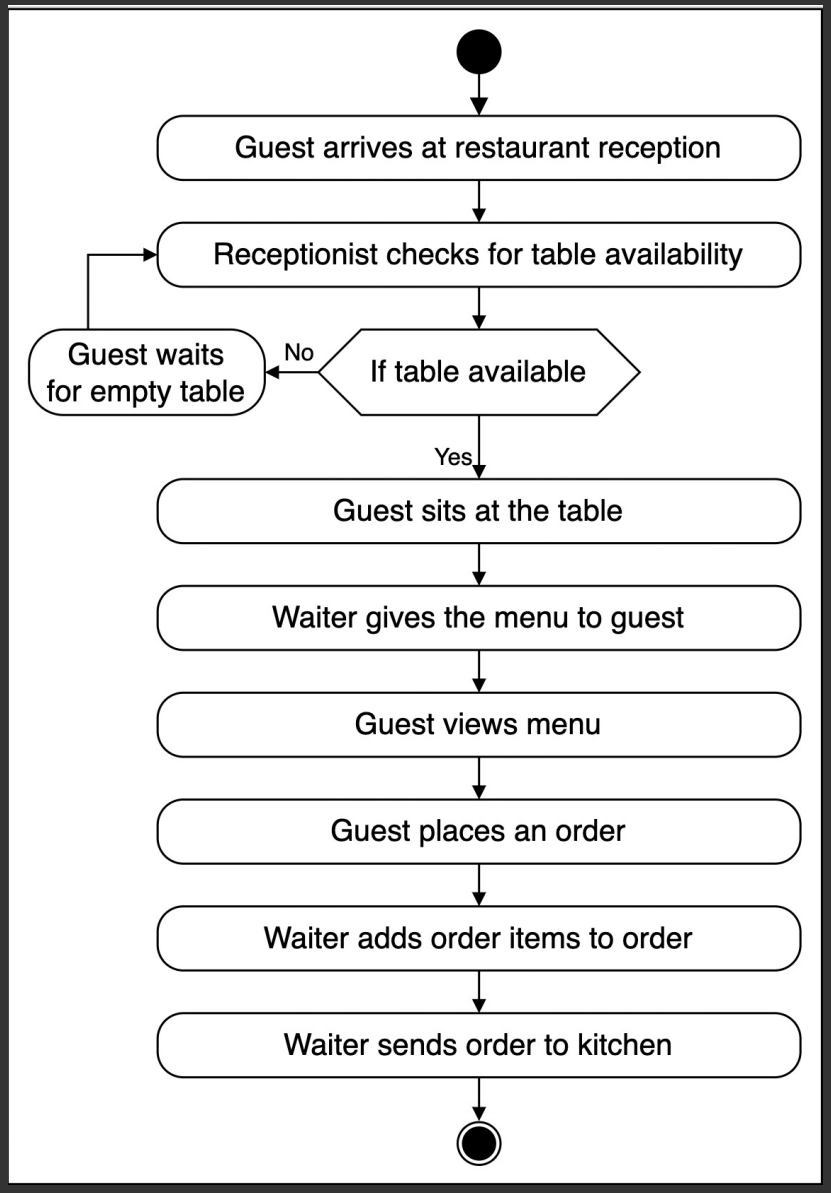




Sequence diagram of modify order



Sequence diagram.
order payment



Activity diagram for placing the Order.

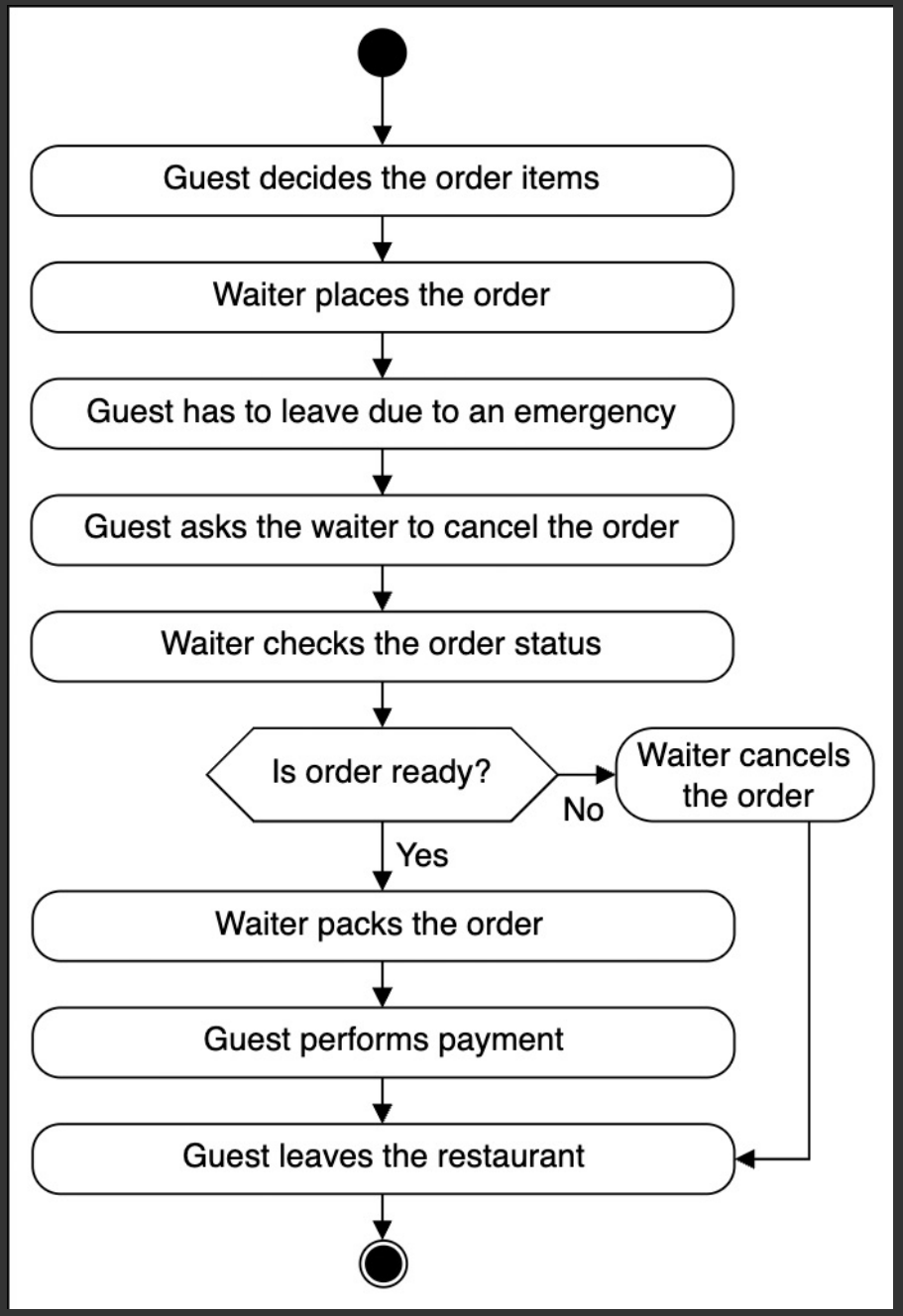


diagram for order cancelation

Code of RMS

2. Account & person

Enumerations.

```
// 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;
}
```

```
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();
}
```

3. Table & Table Set

```
public class Table {
  private int tableID;
  private TableStatus status;
  private int maxCapacity;
  private int locationIdentifier;
  private List<TableSeat> seats;

  public boolean isTableFree();
  public boolean addReservation();
  public static List<Table> search(int capacity, Date startTime);
}

public class TableSeat {
  private int tableSeatNumber;
  private SeatType type;

  public boolean updateSeatType(SeatType type);
}
```

4. Meal & Meal item

```
public class MealItem {
  private int mealItemID;
  private int quantity;
  private MenuItem menuItem;

  public boolean updateQuantity(int
quantity);
}

public class Meal {
  private int mealID;
  private TableSeat seat;
  private List<MenuItem> menuItems;

  public boolean addMealItem(MealItem
mealItem);
}
```

5. Menu, menu Section & menu items

```
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);
}
```

6. Order, Kitchen & Reservation

```
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);
}
```

7. Payment & Bill

```
// 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();
}
```

8. Notification

```
// Notification is an abstract class
public abstract class Notification {
    private int notificationId;
    // The Date data type represents and deals with both date and time.
    private Date createdOn;
    private String content;

    public abstract void send(Person person);
}

class SmsNotification extends Notification {
    private String phone;

    public void sendNotification(Person person) {
        // functionality
    }
}

class EmailNotification extends Notification {
    private String email;

    public void sendNotification(Person person) {
        // functionality
    }
}
```

9. Seating chart, Branch & restaurant

```
public class SeatingChart {
  private int seatingChartID;
  private byte[] seatingChartImage;

  public boolean print();
}

public class Branch {
  private String name;
  private Address location;
  private Kitchen kitchen;
  private Menu menu;

  public Address addseatingChart();
}

public class Restaurant {
  private String name;
  private List<Branch> branches;

  public boolean addBranch(Branch branch);
}
```