UML Class Diagram Design Report

Mohamed Amine EL Kalai

October 16, 2025

Contents

1 Exercise 1: Warm-up

1.1 Exercise 1.1: Library

1.1.1 System Overview

This exercise models a library system that manages books, members, and borrowing records.

1.1.2 Design Decisions

Book Hierarchy:

- Book is the parent class with attribute: name (String)
- Magazine extends Book and adds: edition (int)
- Study Book extends Book and adds: field (String)
- This inheritance structure avoids repeating the name attribute in each book type

Library Collection:

- Library contains multiple books [1..*], meaning at least one book
- Library has a Borrow(Book) operation to handle lending

Borrowing System:

- Member (identified by ID) can borrow multiple books
- Record class tracks each borrowing with borrowDate and returnDate
- The dashed line shows Record connects Member and Book

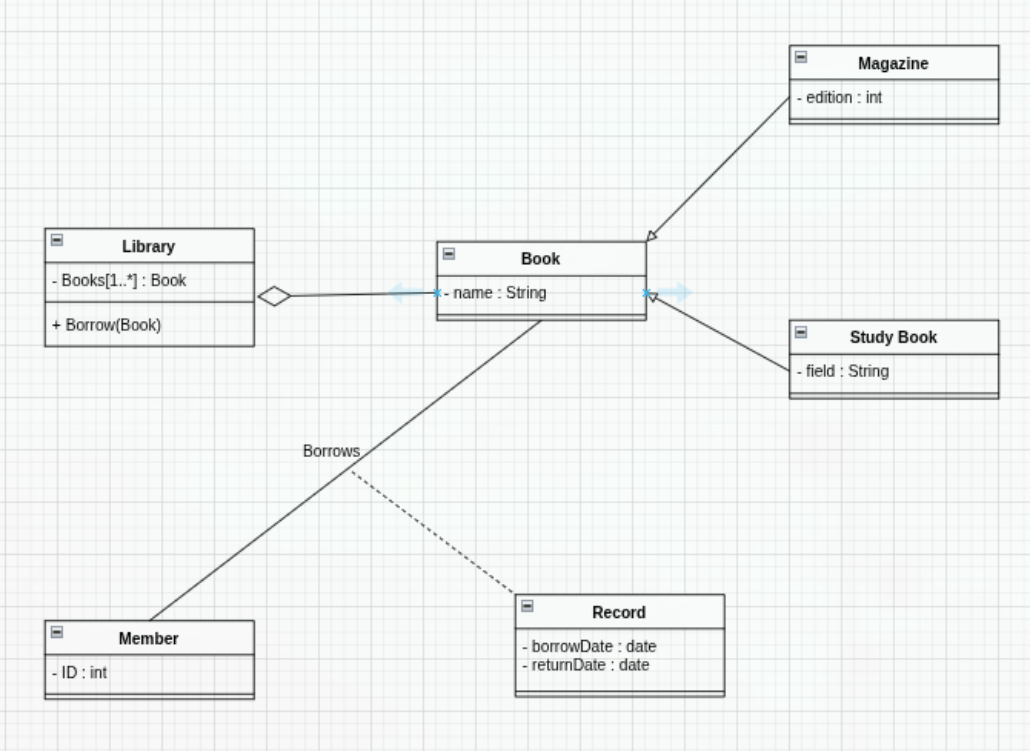

Figure 1: Library Management System Class Diagram

1.2 Exercise 1.2: Laptop

1.2.1 System Overview

This exercise models the relationship between a laptop, keyboard, keys, and owner.

1.2.2 Design Decisions

Composition (Strong Relationship):

- Laptop contains Keyboard - the keyboard cannot exist without the laptop
- Keyboard contains Keys [0..1] - keys are part of the keyboard
- Filled diamonds show these strong ownership relationships

Aggregation (Weak Relationship):

- Laptop has an Owner - but the owner exists independently
- Hollow diamond shows this weaker association

Attributes:

- Laptop: price (int), keyboard reference
- Keyboard: model (String)
- Key: character (char)

- Owner: firstName and lastName (String)

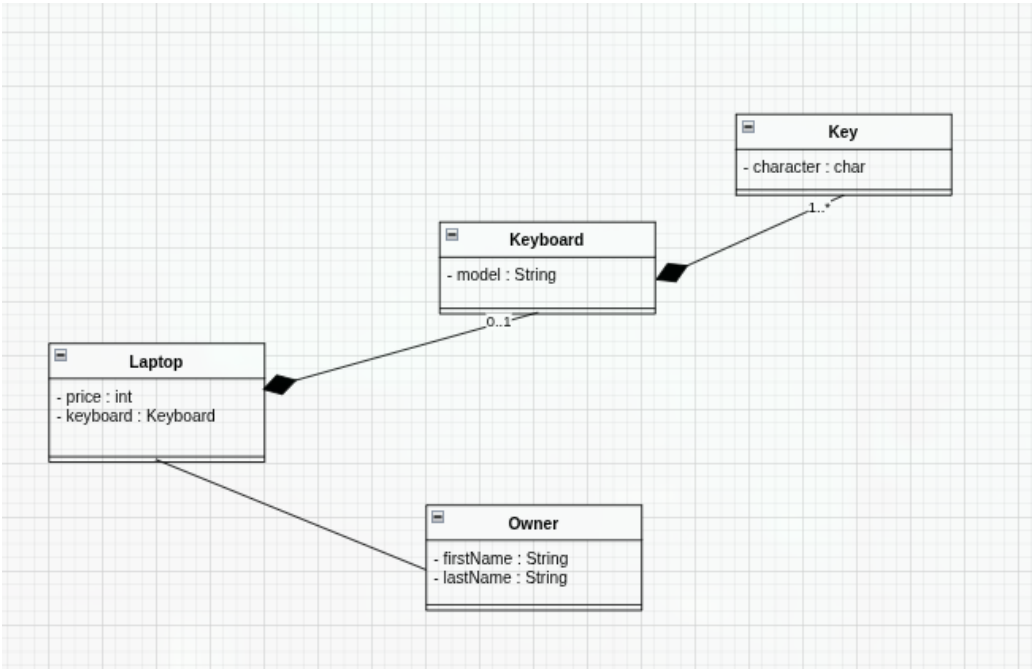

Figure 2: Computer Hardware System Class Diagram

2 Exercise 2: Restaurant

2.1 System Overview

This system models a restaurant with dishes, ingredients, meals, customers, waiters, and tables.

2.2 Design Decisions

Dish Composition:

- Each Dish contains multiple Ingredients [1..*]
- Ingredient has: name, unitOfMeasurement (String), and quantityInStock (double)
- Dish has: ID, name (String), and ingredients list

Meal Structure:

- Meal includes multiple Dishes [1..*]
- Meal tracks: date (Date), startTime and endTime (Time)
- Each Meal occurs at a Table and is served by one Waiter

Restaurant Staff:

- Waiter is responsible for meals [1 waiter per meal]
- Waiter has: ID, name, address, phoneNumber (String)

Customer Tracking:

- Customer can consume meals [0..1] - optional relationship
- Customer has: taxID, name, address (String)

Table Management:

- Table has: ID (String) and maxPeople (int)
- Multiple meals can occur at same table over time

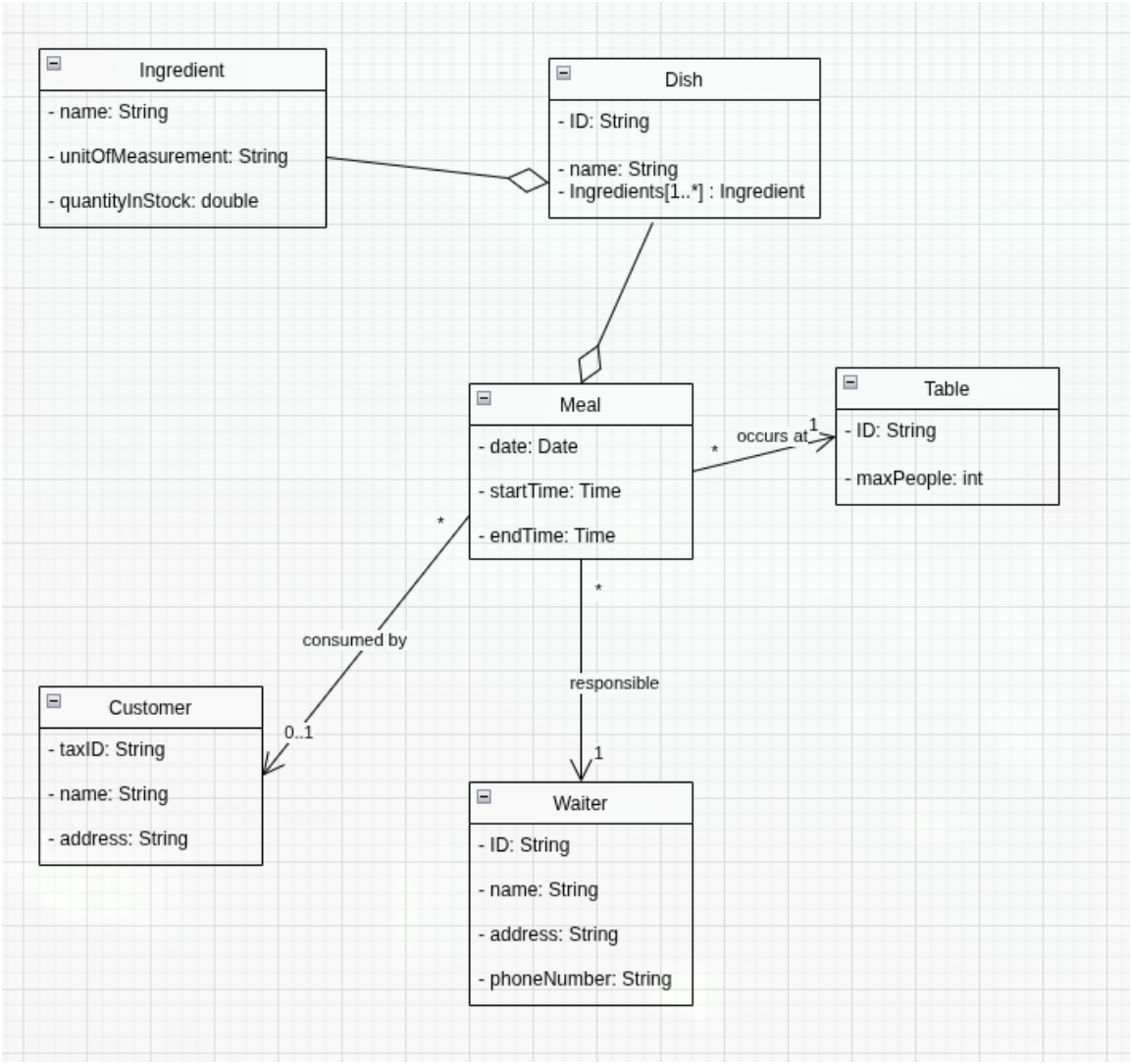

Figure 3: Restaurant Management System Class Diagram

3 Exercise 3: College

3.1 System Overview

This system models an educational institution with students, teachers, departments, colleges, classrooms, subjects, and assessments.

3.2 Design Decisions

Person Inheritance:

- Person is the base class with: lastName, firstName, phone, email (String)
- Student extends Person and adds: entryYear (int)
- Teacher extends Person and adds: startDate (Date)
- This avoids repeating common personal information

Institutional Structure:

- College (with name and website) contains Departments [1..*]
- Department has teachers teaching in it
- Each Department has one Head (a Teacher)

Teaching Relationships:

- Teachers teach multiple Subjects
- Subjects take place in Classrooms (with numSeats attribute)
- Multiple subjects can use the same classroom

Assessment System:

- Students complete Assessments
- Assessments are linked to Subjects
- Assessment stores: grade (float)

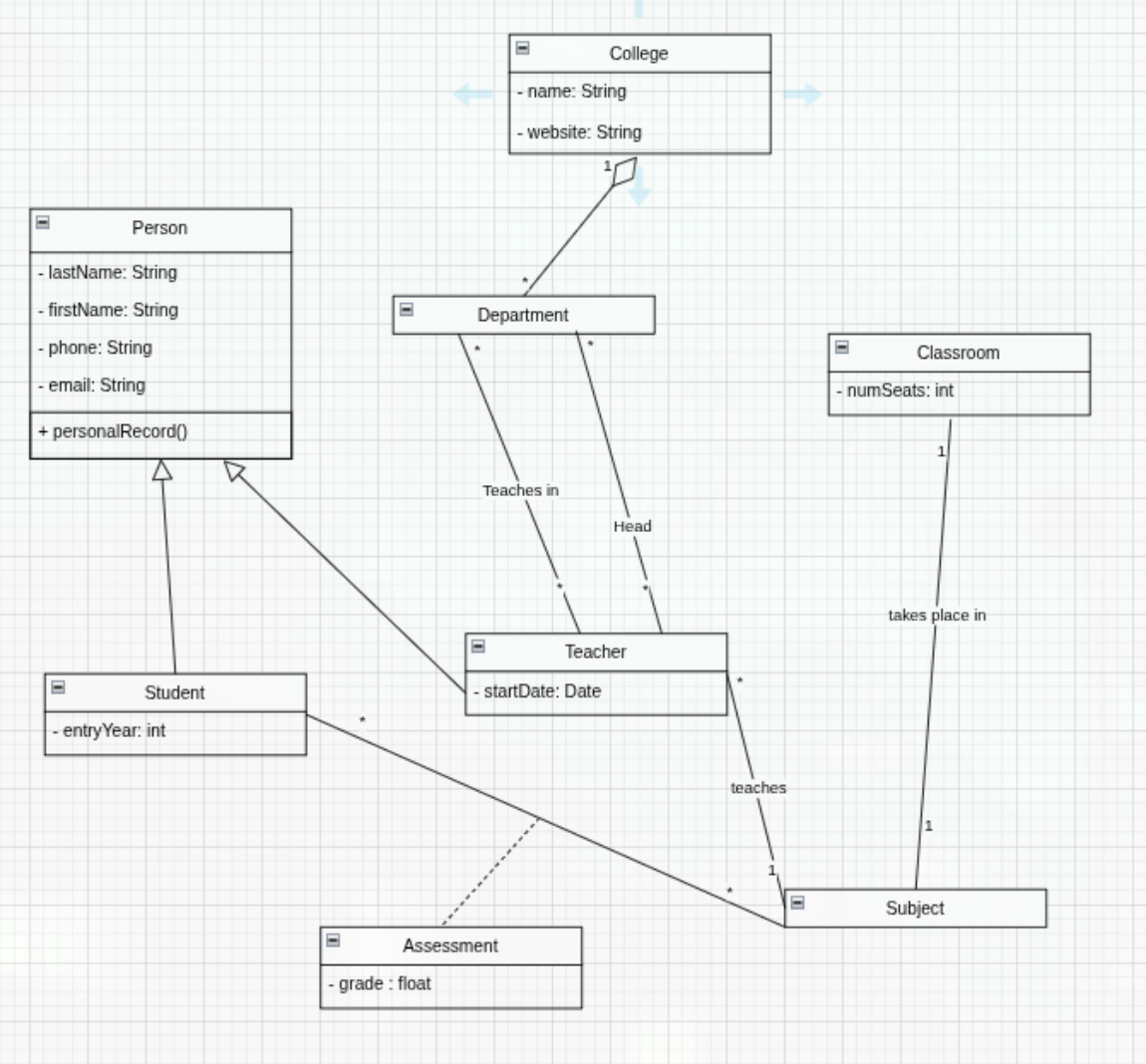

Figure 4: Educational Institution System Class Diagram

4 Exercise 4: Airline Management System

4.1 System Overview

This system models flights, airlines, passengers, reservations, airports, cities, and stops.

4.2 Design Decisions

Flight Operations:

- Airline offers Flights [1..*]
- Flight has: departureDate, arrivalDate (date), departureTime, arrivalTime (time)
- Client can book multiple Flights
- Passenger (with ID) can be on multiple Flights [1..*]

Reservation System:

- Reservation links Passengers to Flights
- Reservation has: isConfirmed (boolean) to track booking status
- A reservation can include multiple flights [2..*] - for round trips or multi-city
- Passengers can have multiple reservations

Geographic Structure:

- City (identified by Code) is served by multiple Airports [1..*]
- Airport has field attributes and methods
- Each Airport can serve multiple cities

Stop Abstraction (Important Design Choice):

- Stop class represents departure, arrival, and stopover points
- Stop has: stopOrder (int), departureTime (time), arrivalTime (time)
- Each Stop is located at an Airport [1]
- Each Flight has multiple Stops [2..*]

Key Design Principle:

The departure, arrival and stopover stations are considered the same class "Stop" (due to abstraction principle) since they have the same attributes (arrivalTime, departureTime, airport). This helps optimize the class diagram and make it even better. However, a mandatory change must be added: each flight has to have at least 2 stops (departure and arrival).

This abstraction simplifies the model by avoiding three separate classes that would have identical attributes. The stopOrder attribute allows proper sequencing of the flight route.

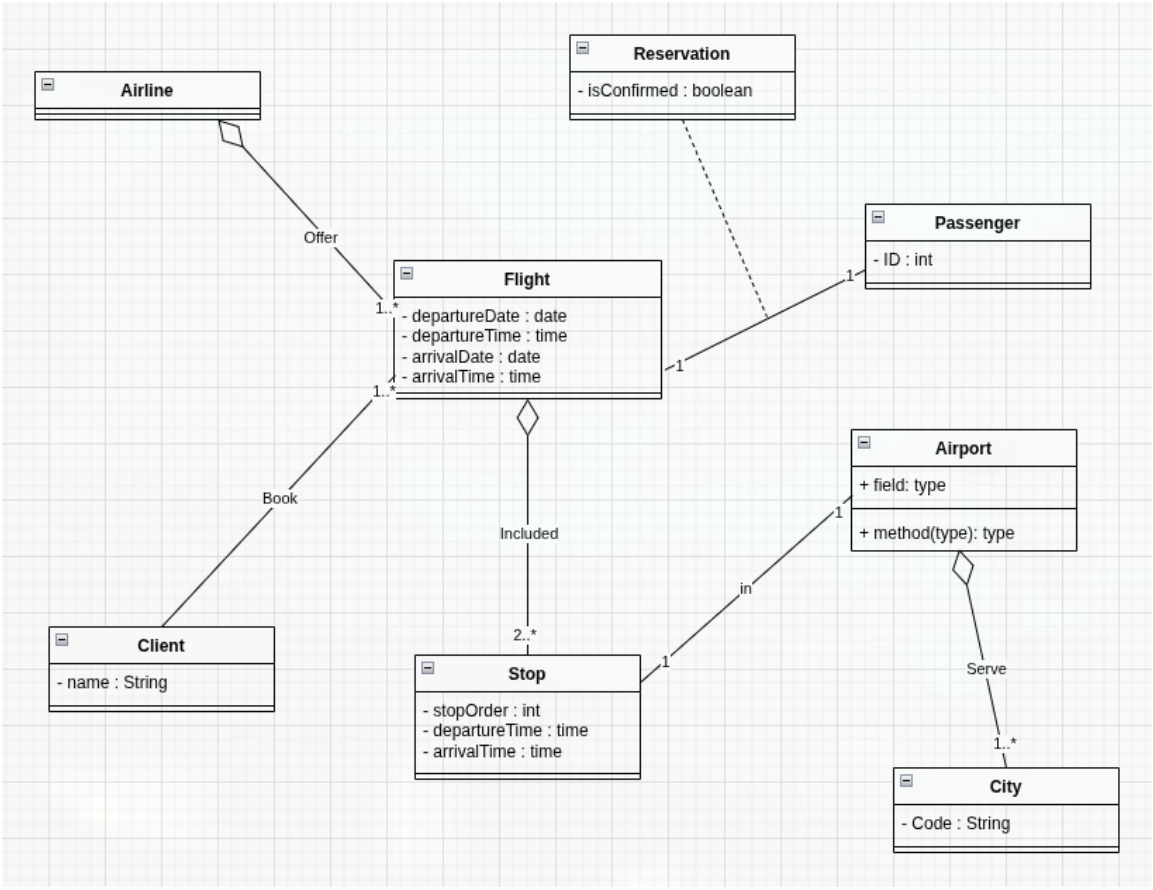

Figure 5: Flight Reservation System Class Diagram

5 Conclusion

This report presented four UML class diagram exercises demonstrating key object-oriented concepts:

- **Inheritance:** Used in library books, educational persons, reducing repeated attributes
- **Composition vs Aggregation:** Shown in laptop-keyboard (strong) vs laptop-owner (weak)
- **Association Classes:** Record class linking members and books with temporal data
- **Abstraction:** Stop class unifying departure, arrival, and stopover stations
- **Cardinalities:** Enforcing business rules like minimum stops per flight