Object Modeling: Object Relationships and Communication

Problem 1: Library and Books (Aggregation)

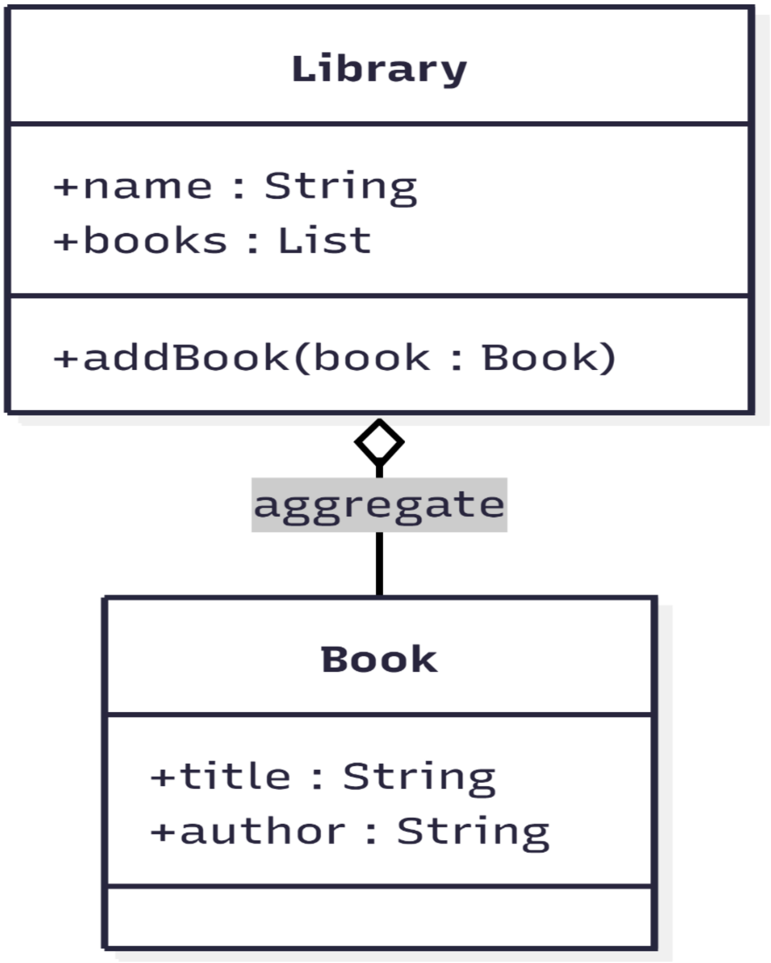
Description: Create a Library class that contains multiple Book objects. Model the relationship such that a library can have many books, but a book can exist independently (outside of a specific library).

Tasks:

- Define a Library class with a List<Book> collection.
- Define a Book class with attributes such as Title and Author.
- Demonstrate the aggregation relationship by creating books and adding them to different libraries.

Goal: Understand aggregation by modeling a real-world relationship where the Library aggregates Book objects.

Sol :



Problem 2: Bank and Account Holders (Association)

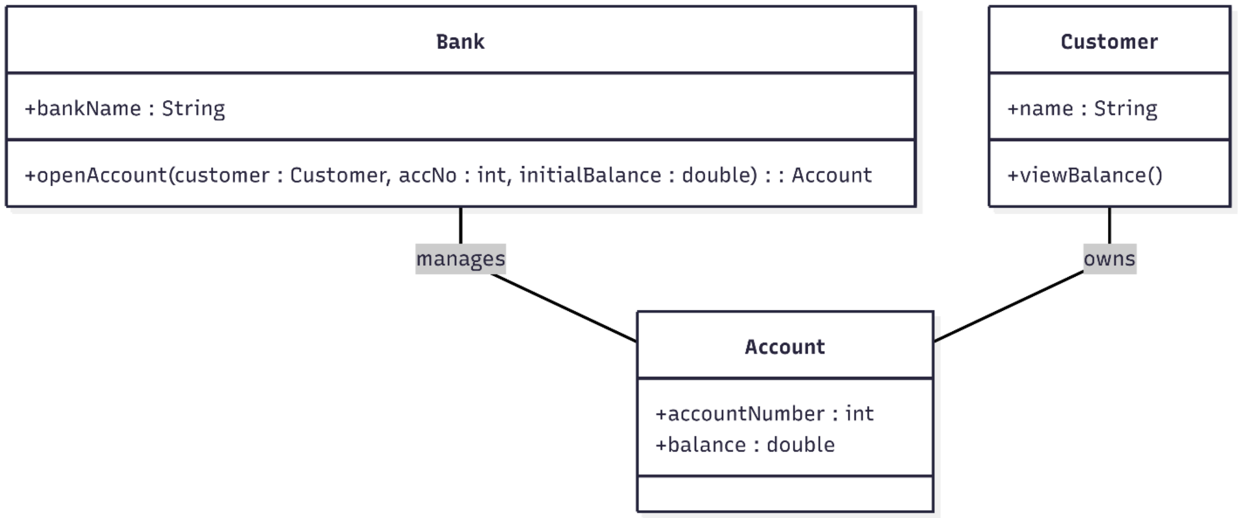
Description: Model a relationship where a Bank has Customer objects associated with it. A Customer can have multiple bank accounts, and each account is linked to a Bank.

Tasks:

- Define a Bank class and a Customer class.
- Use an association relationship to show that each Customer has an account in a Bank.
- Implement methods that enable communication, such as OpenAccount() in the Bank class and ViewBalance() in the Customer class.

Goal: Illustrate association by setting up a relationship between customers and the bank.

Sol:



Problem 3: Company and Departments (Composition)

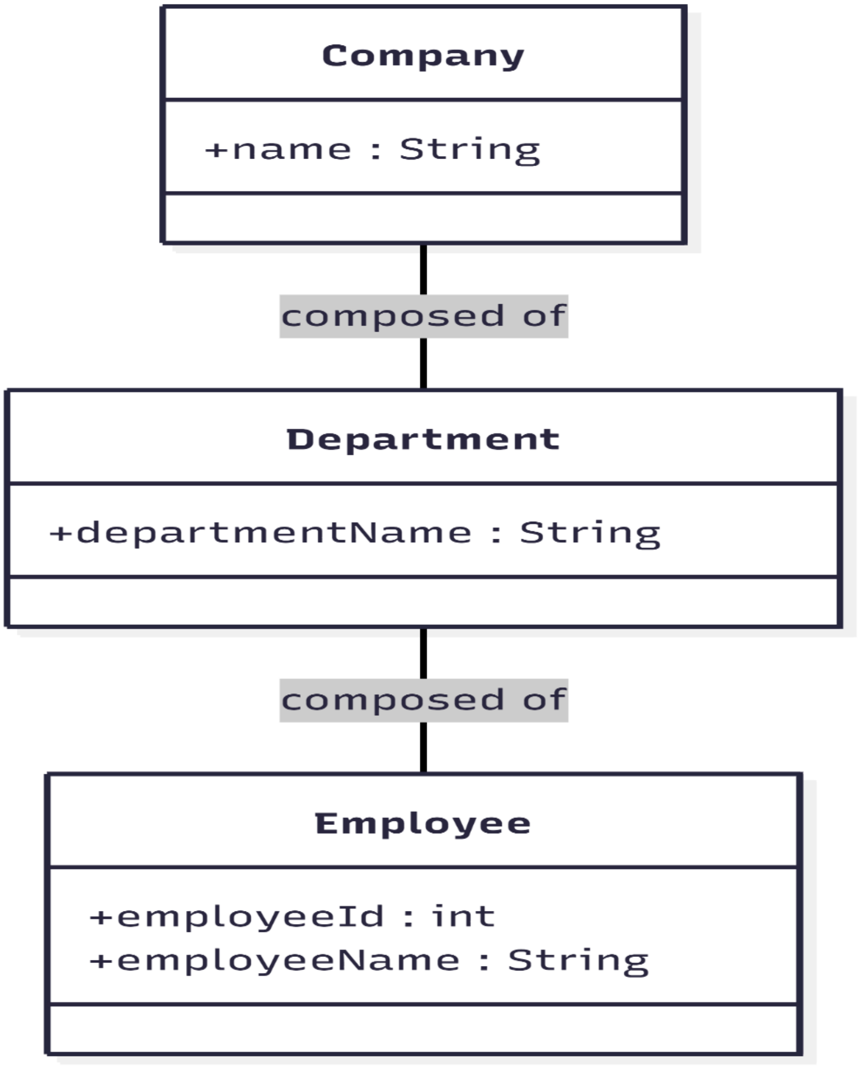
Description: A Company has several Department objects, and each department contains Employee objects. Model this using composition, where deleting a Company should also delete all departments and employees.

Tasks:

- Define a Company class that contains multiple Department objects.
- Define an Employee class within each Department.
- Show the composition relationship by ensuring that when a Company object is deleted, all associated Department and Employee objects are also removed.

Goal: Understand composition by implementing a relationship where Department and Employee objects cannot exist without a Company.

Sol:



Problem 4: School and Students with Courses (Association and Aggregation)

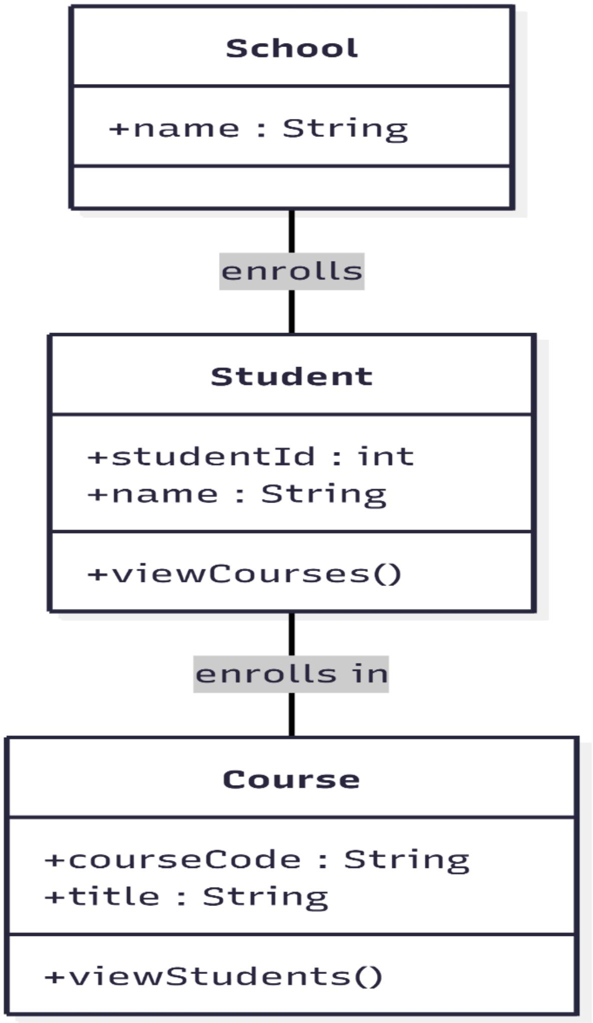
Description: Model a School with multiple Student objects, where each student can enroll in multiple courses, and each course can have multiple students.

Tasks:

- Define School, Student, and Course classes.
- Model an association between Student and Course to show that students can enroll in multiple courses.
- Model an aggregation relationship between School and Student.
- Demonstrate how a student can view the courses they are enrolled in and how a course can show its enrolled students.

Goal: Practice association by modeling many-to-many relationships between students and courses.

Sol :



Problem 5: University with Faculties and Departments (Composition and Aggregation)

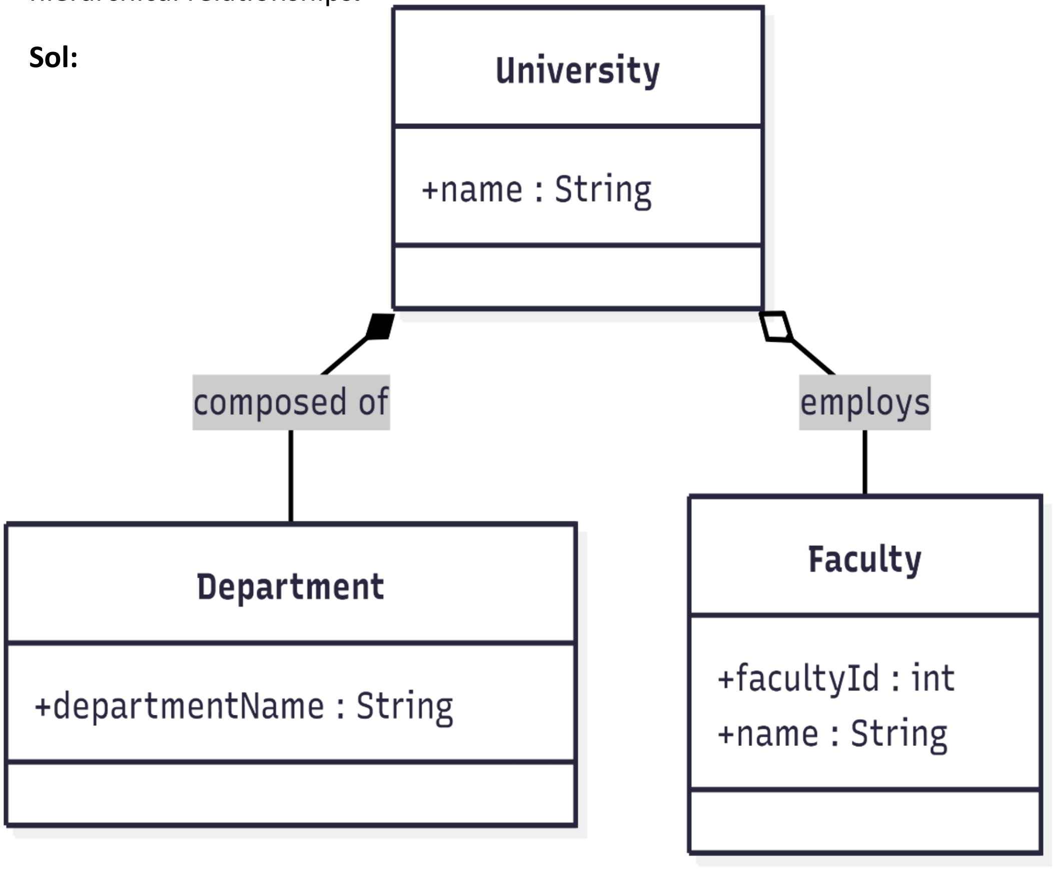
Description: Create a University with multiple Faculty members and Department objects. Model it so that the University and its Departments are in a composition relationship (deleting a university deletes all departments), and the Faculty members are in an aggregation relationship (faculty can exist outside of any specific department).

Tasks:

- Define a University class with Department and Faculty classes.
- Demonstrate how deleting a University also deletes its Departments.
- Show that Faculty members can exist independently of a Department.

Goal: Understand the differences between composition and aggregation in modeling complex hierarchical relationships**.**

Sol:



Problem 6 : Hospital, Doctors, and Patients (Association and Communication)

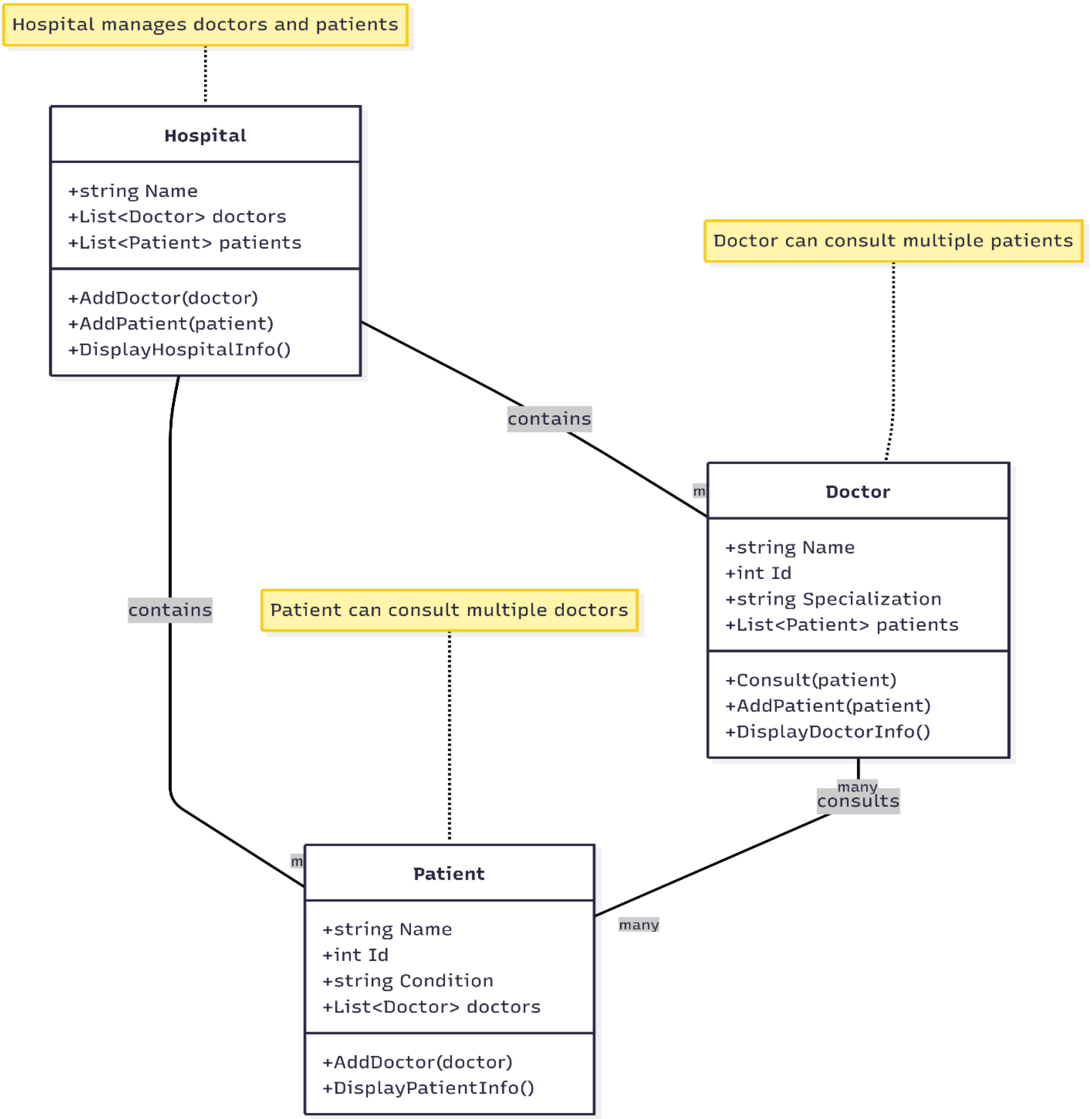
Description: Model a Hospital where Doctor and Patient objects interact through consultations. A doctor can see multiple patients, and each patient can consult multiple doctors**.**

Tasks:

- Define a Hospital class containing Doctor and Patient classes.
- Create a method Consult() in the Doctor class to show communication, which would display the consultation between a doctor and a patient.
- Model an association between doctors and patients to show that doctors and patients can have multiple relationships.

Goal: Practice creating an association with communication between objects by modeling doctor-patient consultations.

Sol:

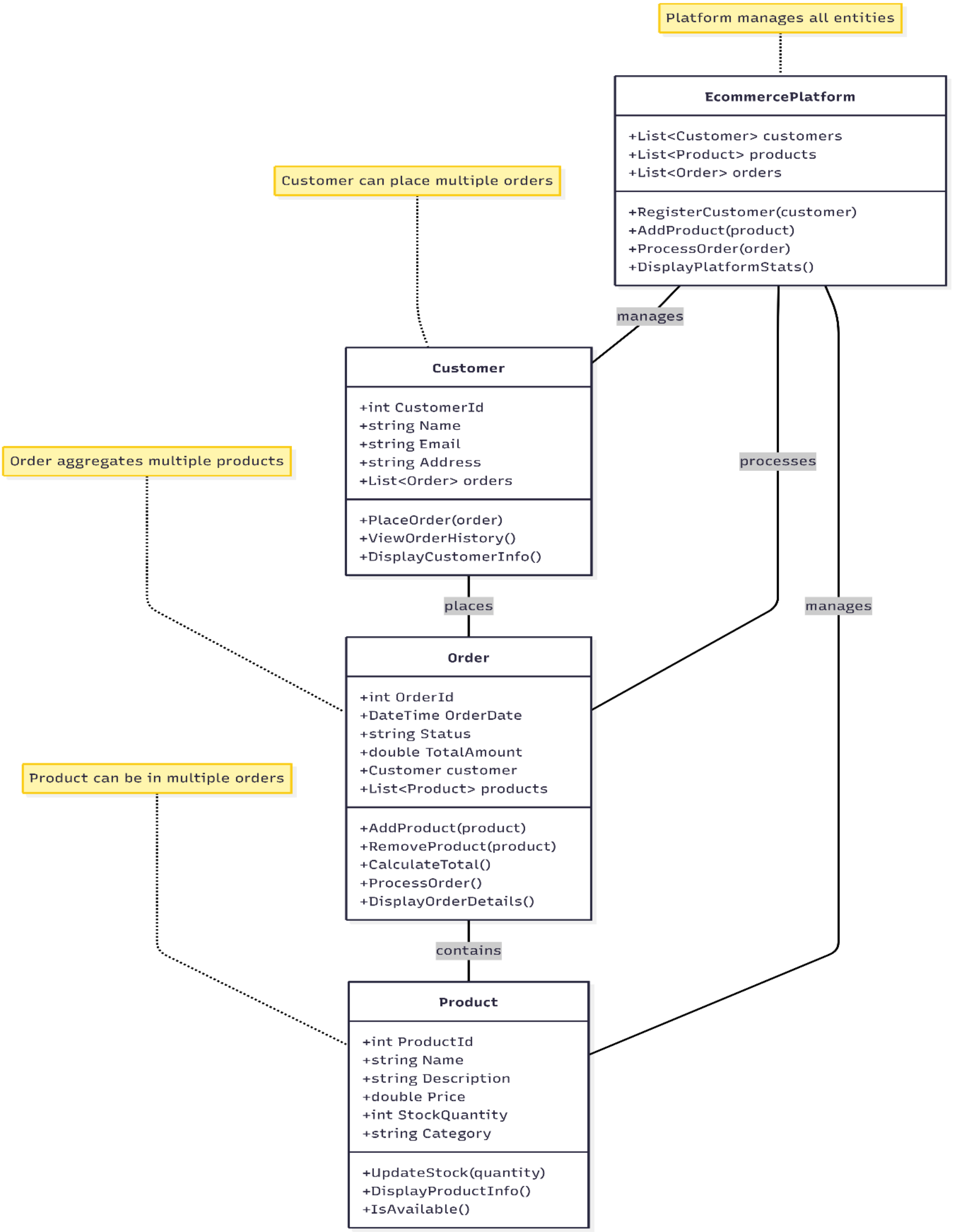


Problem 7 : E-commerce Platform with Orders, Customers, and Products

Description: Design an e-commerce platform with Order, Customer, and Product classes. Model relationships where a Customer places an Order, and each Order contains multiple Product objects.

Goal: Show communication and object relationships by designing a system where customers communicate through orders, and orders aggregate products.

sol:

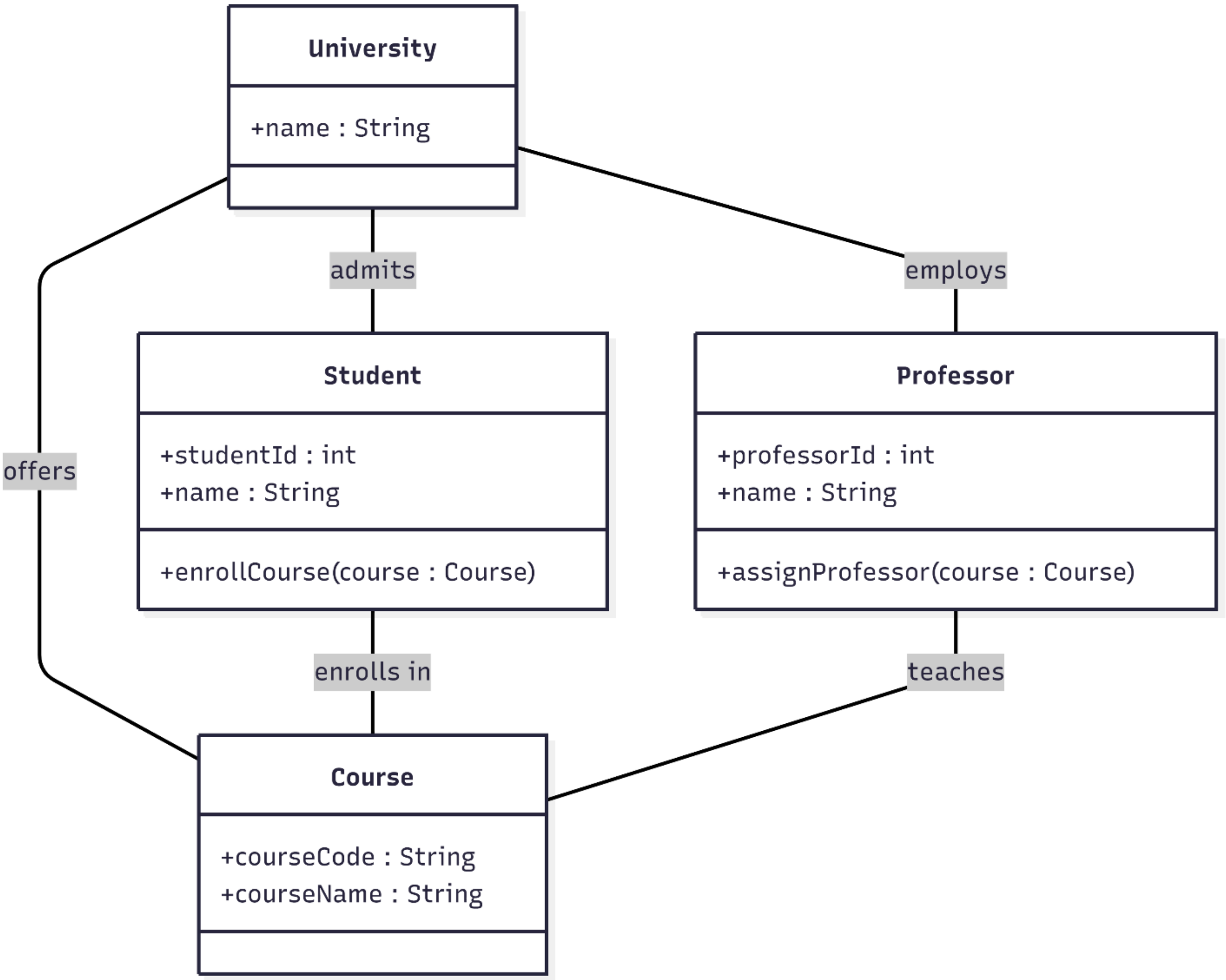


Problem 8 : University Management System

Description: Model a university system with Student, Professor, and Course classes. Students enroll in courses, and professors teach courses. Ensure students and professors can communicate through methods like EnrollCourse() and AssignProfessor().

Goal: Use association and aggregation to create a university system that emphasizes relationships and interactions among students, professors, and courses.

Sol:


University
+name : String
admits
employs
offers
Student
+studentId : int
+name : String
+enrollCourse(course : Course)
Professor
+professorId : int
+name : String
+assignProfessor(course : Course)
enrolls in
teaches
Course
+courseCode : String
+courseName : String