Bài 1:

**1. Class Sach (Book)**

**Vai trò:** Đại diện cho thông tin của một cuốn sách trong thư viện.  
**Thuộc tính:**

* maSach: string — Mã định danh duy nhất của sách
* tenSach: string — Tên sách
* tacGia: string — Tác giả
* namXuatBan: int — Năm xuất bản
* tinhTrang: boolean — Trạng thái (true = còn trong thư viện, false = đã được mượn)

**Phương thức:**

* hienThiThongTin(): hiển thị thông tin chi tiết sách
* capNhatTinhTrang(): cập nhật trạng thái khi mượn hoặc trả

**2. Class DocGia (Reader)**

**Vai trò:** Quản lý thông tin của người đọc/mượn sách.  
**Thuộc tính:**

* maDocGia: string — Mã độc giả
* tenDocGia: string — Họ tên độc giả
* soDienThoai: string — Số điện thoại
* diaChi: string — Địa chỉ liên hệ

**Phương thức:**

* dangKy(): đăng ký trở thành độc giả
* muonSach(Sach s): thực hiện mượn sách
* traSach(Sach s): thực hiện trả sách

**3. Class NhanVien (Staff)**

**Vai trò:** Đại diện cho nhân viên thư viện – người quản lý hoạt động mượn/trả sách.  
**Thuộc tính:**

* maNhanVien: string — Mã nhân viên
* tenNhanVien: string — Họ tên nhân viên
* chucVu: string — Chức vụ

**Phương thức:**

* quanLyMuonTra(): xác nhận và ghi nhận các giao dịch mượn/trả sách
* themSachMoi(): thêm sách mới vào hệ thống

**4. Class PhieuMuon (BorrowSlip / LoanRecord)**

**Vai trò:** Ghi lại thông tin mỗi lần mượn/trả sách.  
**Thuộc tính:**

* maPhieu: string — Mã phiếu mượn
* docGia: DocGia — Thông tin người mượn
* sach: Sach — Thông tin sách được mượn
* ngayMuon: Date — Ngày mượn
* ngayTra: Date — Ngày trả (nếu có)
* tinhTrang: string — “Đang mượn” / “Đã trả”

**Phương thức:**

* xacNhanMuon(): xác nhận việc mượn
* xacNhanTra(): cập nhật khi trả sách

**5. Class ThuVien (Library)**

**Vai trò:** Lớp tổng quản lý hệ thống, lưu danh sách sách, độc giả và nhân viên.  
**Thuộc tính:**

* danhSachSach: List<Sach>
* danhSachDocGia: List<DocGia>
* danhSachNhanVien: List<NhanVien>
* danhSachPhieuMuon: List<PhieuMuon>

**Phương thức:**

* timSachTheoTen(ten): tìm kiếm sách
* themDocGia(docGia): thêm độc giả mới
* xoaSach(maSach): xóa sách khỏi hệ thống

**Tổng kết mối quan hệ:**

* DocGia ↔ PhieuMuon: 1 độc giả có thể có nhiều phiếu mượn
* Sach ↔ PhieuMuon: 1 sách có thể xuất hiện trong nhiều phiếu (theo thời gian)
* NhanVien quản lý hoạt động mượn/trả (qua PhieuMuon)
* ThuVien là lớp tổng hợp (aggregate) quản lý toàn bộ dữ liệu

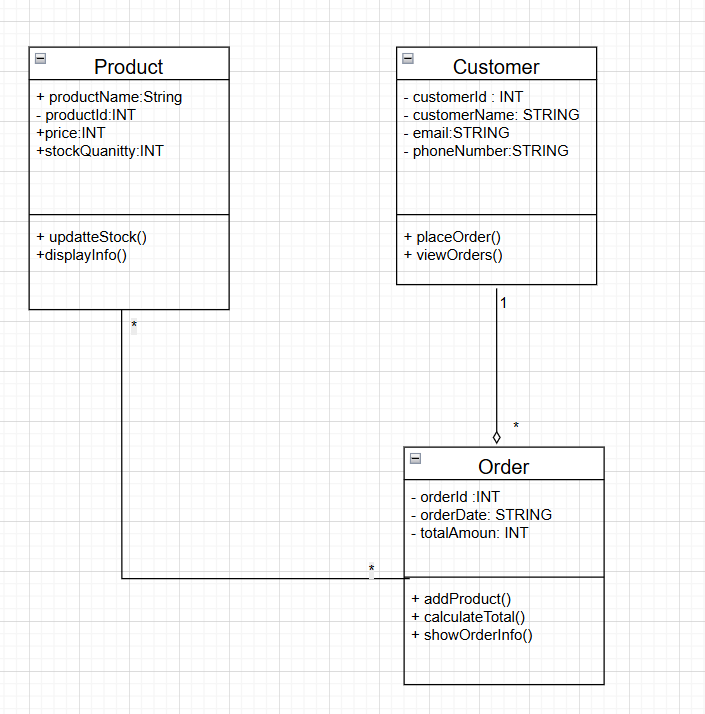
Bài 2:

|  |  |  |  |
| --- | --- | --- | --- |
| **Mối quan hệ** | **Loại quan hệ** | **Giải thích** | **Ký hiệu UML** |
| Một **giáo viên** giảng dạy **nhiều lớp học** | **Association (liên kết)** | Đây là mối quan hệ thông thường giữa hai lớp độc lập: “Giáo viên” có thể giảng dạy nhiều “Lớp học”, và một “Lớp học” có thể có nhiều “Giáo viên”. Không ràng buộc về vòng đời. | Đường nối có thể kèm nhãn “1..*” hoặc “*..\*” giữa hai lớp |
| Một **đơn hàng** gồm **nhiều sản phẩm** | **Aggregation (kết hợp)** | Đây là mối quan hệ “có chứa nhưng có thể tách rời”: “Sản phẩm” vẫn tồn tại độc lập dù không thuộc đơn hàng nào. Khi xóa đơn hàng, sản phẩm không bị xóa. | Hình thoi rỗng (◊) ở phía “Đơn hàng” → “Sản phẩm” |
| Một **cơ thể** gồm **nhiều bộ phận không thể tách rời** | **Composition (thành phần)** | Đây là mối quan hệ “chứa – không thể tách rời”: “Bộ phận” không thể tồn tại nếu “Cơ thể” bị hủy. Khi đối tượng “Cơ thể” bị xóa, các “Bộ phận” cũng bị xóa theo. | Hình thoi đặc (◆) ở phía “Cơ thể” → “Bộ phận” |

Bài 3:

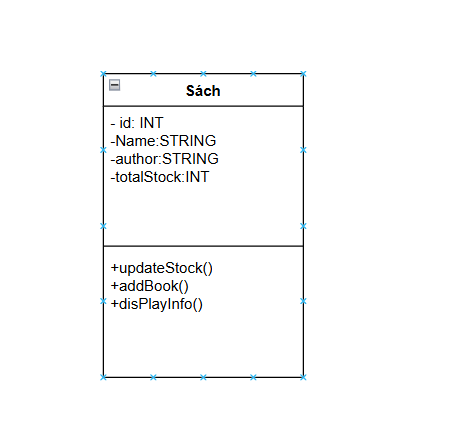
|  |  |  |
| --- | --- | --- |
| **Thành phần** | **Modifier (Phạm vi truy cập)** | **Giải thích** |
| username | private | Thông tin tên đăng nhập chỉ nên được truy cập gián tiếp qua phương thức (getter/setter). Tránh bị thay đổi trực tiếp từ bên ngoài. |
| password | private | Là dữ liệu nhạy cảm, cần được bảo mật tuyệt đối. Chỉ được truy cập bên trong class hoặc thông qua các phương thức kiểm soát như login() hoặc resetPassword(). |
| lastLoginTime | private | Dữ liệu nội bộ dùng để lưu thời điểm đăng nhập cuối cùng, không nên cho phép thay đổi trực tiếp từ bên ngoài. Có thể cung cấp phương thức getLastLoginTime() để xem. |
| login() | public | Phương thức chính để người dùng bên ngoài (ứng dụng, giao diện) gọi thực hiện chức năng đăng nhập. |
| resetPassword() | public | Cần được gọi từ bên ngoài khi người dùng quên mật khẩu hoặc quản trị viên yêu cầu đổi mật khẩu. |

Bài 4:

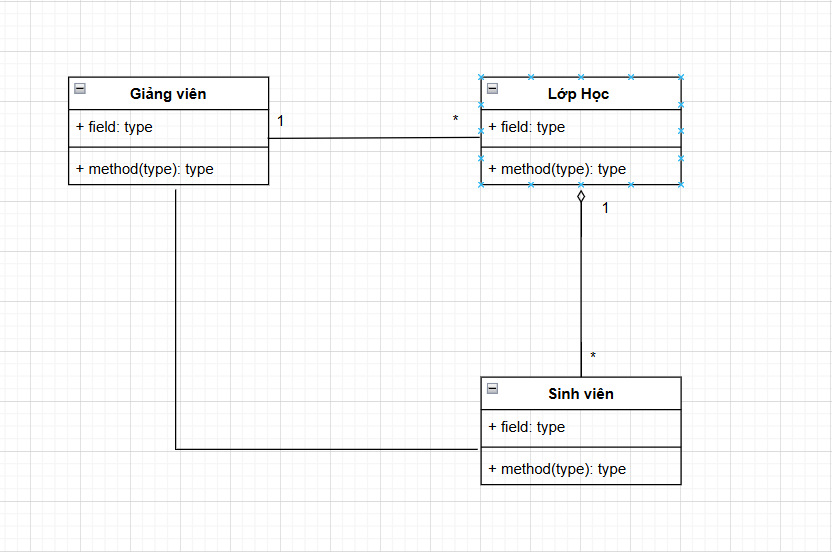


|  |  |  |
| --- | --- | --- |
| Thực thể | Quan hệ trực tiếp với | Giải thích |
| Customer | Order | Mỗi khách hàng có thể tạo ra nhiều đơn hàng. Đây là quan hệ 1–n (một-nhiều). |
| Order | Product | Mỗi đơn hàng có thể chứa nhiều sản phẩm, và một sản phẩm có thể nằm trong nhiều đơn hàng -> quan hệ n–n (nhiều-nhiều). |
| Customer | *(Không trực tiếp)* | Khách hàng không trực tiếp “sở hữu” hay “chứa” sản phẩm. Việc mua sản phẩm chỉ diễn ra thông qua đơn hàng. |

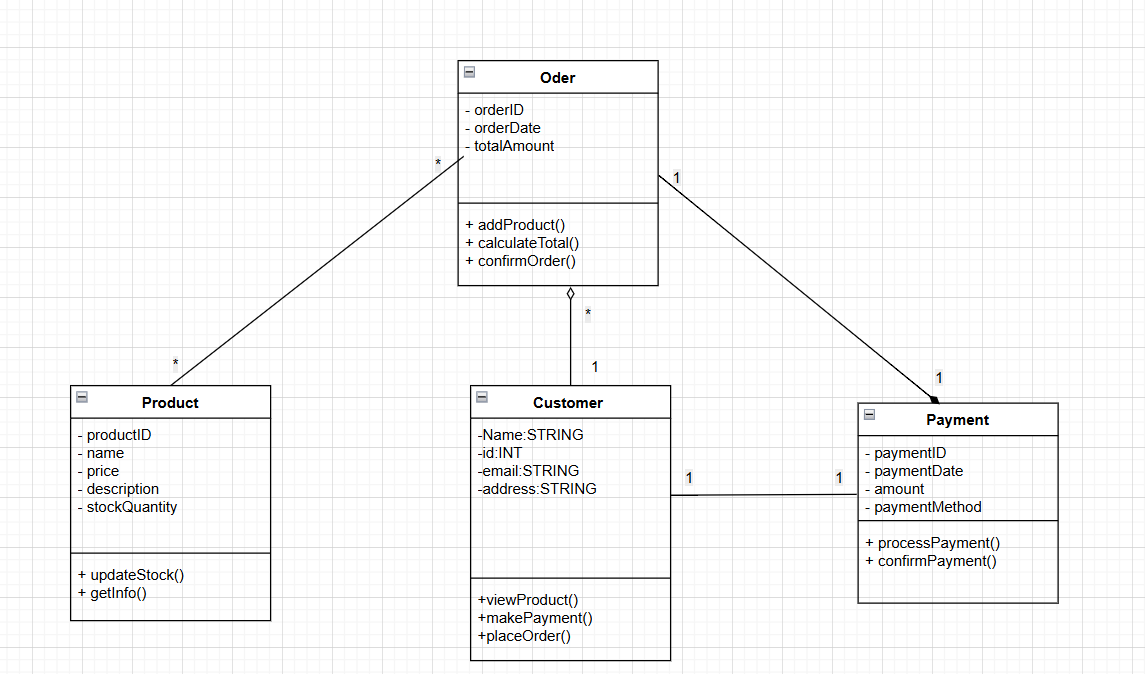
Bài 5:



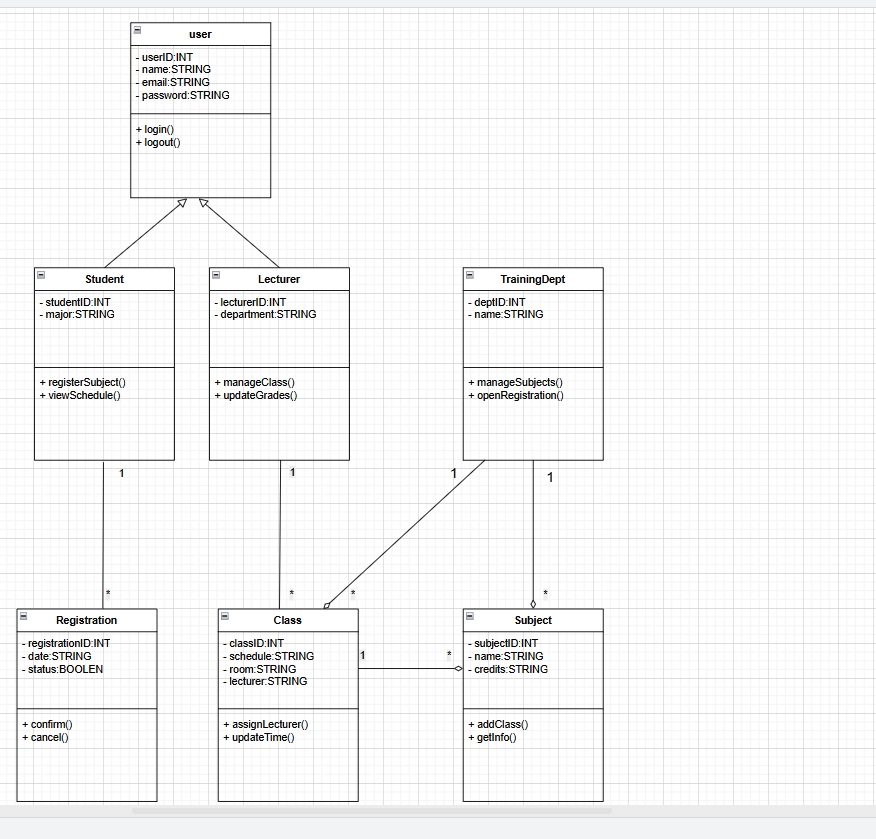
Bài 6:



Bài 7:



Bài 8:



Bài 9:

1. Lỗi quan hệ giữa các lớp (Relationship)

Hiện sơ đồ vẽ User 1—1 Order, tức là một người dùng chỉ có một đơn hàng -> sai thực tế.  
Thực tế: một User có thể có nhiều Order, nhưng mỗi Order chỉ thuộc về 1 User.  
Sửa lại: User 1 —— \* Order

2. Lỗi multiplicity (Bội số quan hệ)

Như phân tích ở trên, multiplicity “1—1” sai.  
Sửa lại: User (1) —— (0..\*) Order

Giải thích: Một người dùng có thể có nhiều đơn hàng hoặc chưa có đơn nào.

3. Lỗi modifier (Truy cập dữ liệu)

Hiện tại tất cả thuộc tính đều là + (public) -> không đúng nguyên tắc hướng đối tượng.Sửa lại:

Thuộc tính nên để private (-)

Phương thức nên để public (+)  
Ví dụ:

- username: String

- password: String

- email: String

+ login()

+ resetPassword()

