BÁO CÁO THỰC HÀNH

Lab 2. Ứng dụng UML trong thiết kế phần mềm với PlantUML

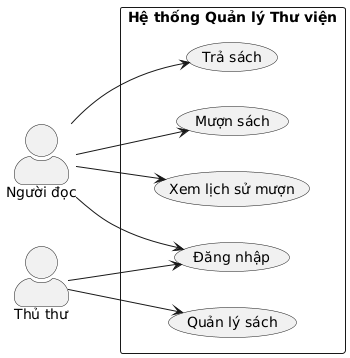
*Họ và tên SV:*

*MSV:*

*Lớp:*

# Task 1. Use case diagram

## Sơ đồ



## Script PlantUML

@startuml

left to right direction

skinparam actorStyle awesome

skinparam packageStyle rectangle

rectangle "Hệ thống Quản lý Thư viện" as System {

usecase "Trả sách" as UC\_Tra

usecase "Mượn sách" as UC\_Muon

usecase "Xem lịch sử mượn" as UC\_LS

usecase "Đăng nhập" as UC\_Login

usecase "Quản lý sách" as UC\_QL

}

actor "Người đọc" as Reader

actor "Thủ thư" as Librarian

Reader --> UC\_Tra

Reader --> UC\_Muon

Reader --> UC\_LS

Reader --> UC\_Login

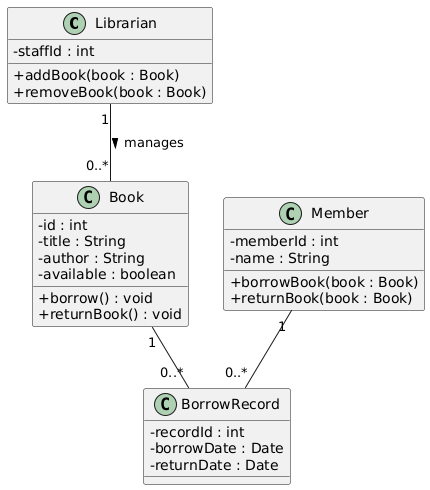
Librarian --> UC\_Login

Librarian --> UC\_QL

@enduml

# Task 2. Class diagram

## Sơ đồ



## Script PlantUML

@startuml

skinparam classAttributeIconSize 0

class Librarian {

- staffId : int

+ addBook(book : Book)

+ removeBook(book : Book)

}

class Member {

- memberId : int

- name : String

+ borrowBook(book : Book)

+ returnBook(book : Book)

}

class Book {

- id : int

- title : String

- author : String

- available : boolean

+ borrow() : void

+ returnBook() : void

}

class BorrowRecord {

- recordId : int

- borrowDate : Date

- returnDate : Date

}

' Quan hệ

Librarian "1" -- "0..\*" Book : manages >

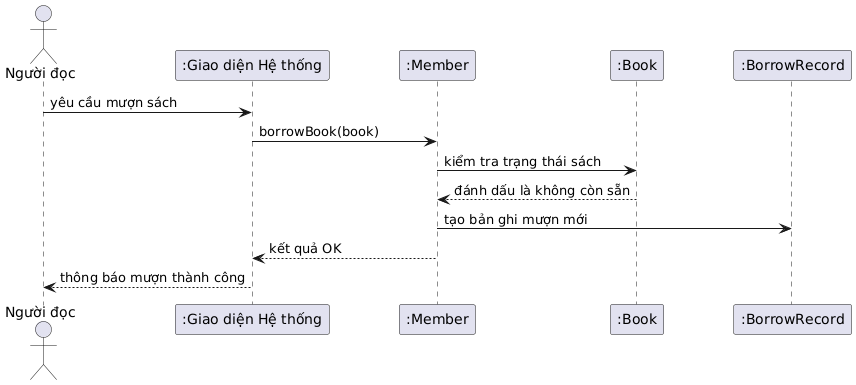
Member "1" -- "0..\*" BorrowRecord

Book "1" -- "0..\*" BorrowRecord

@enduml

# Task 3. Sequence diagram

## Sơ đồ



## Script PlantUML

@startuml

skinparam sequenceArrowThickness 1

skinparam participantPadding 30

skinparam BoxPadding 10

skinparam responseMessageBelowArrow true

actor "Người đọc" as Reader

boundary "Giao diện Hệ thống" as UI

control "Member" as Member

entity "Book" as Book

database "BorrowRecord" as BR

Reader -> UI : yêu cầu mượn sách

UI -> Member : borrowBook(book)

Member -> Book : kiểm tra trạng thái sách

alt Sách có sẵn

Book --> Member : available = true

Member -> Book : đánh dấu là không còn sẵn

Member -> BR : tạo bản ghi mượn mới

Member --> UI : OK

UI --> Reader : thông báo mượn thành công

else Hết sách

Book --> Member : available = false

Member --> UI : lỗi: sách không còn sẵn

UI --> Reader : thông báo mượn thất bại

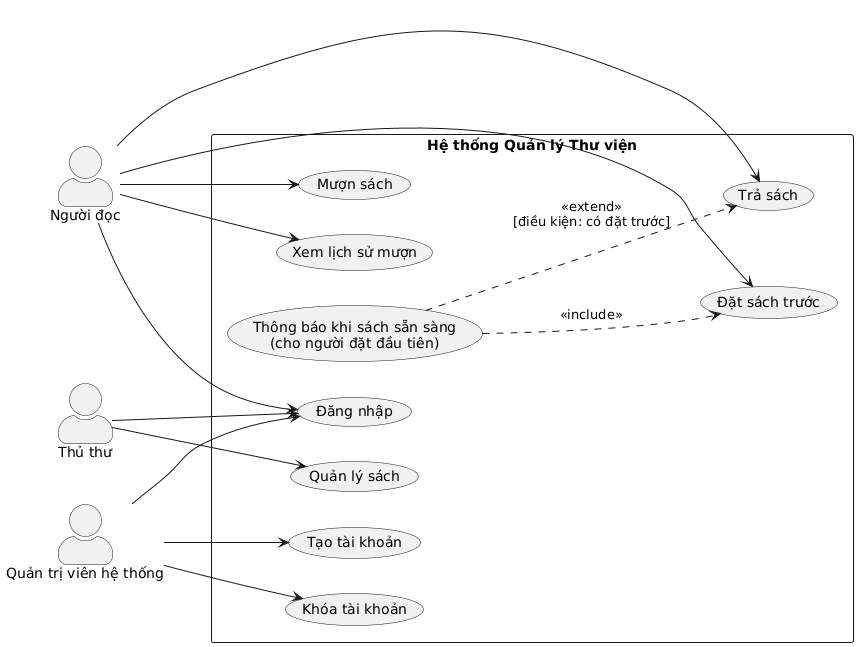
end

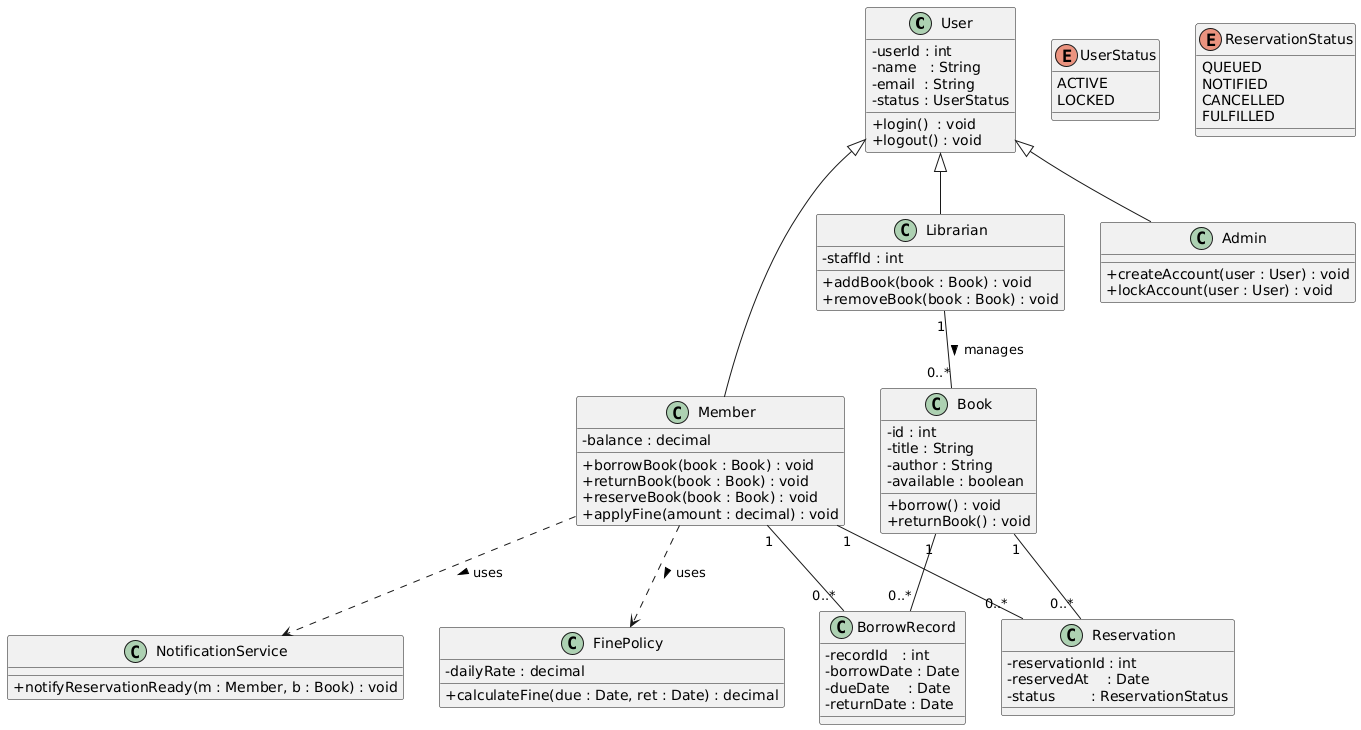
@enduml

# Task 4. Mở rộng yêu cầu

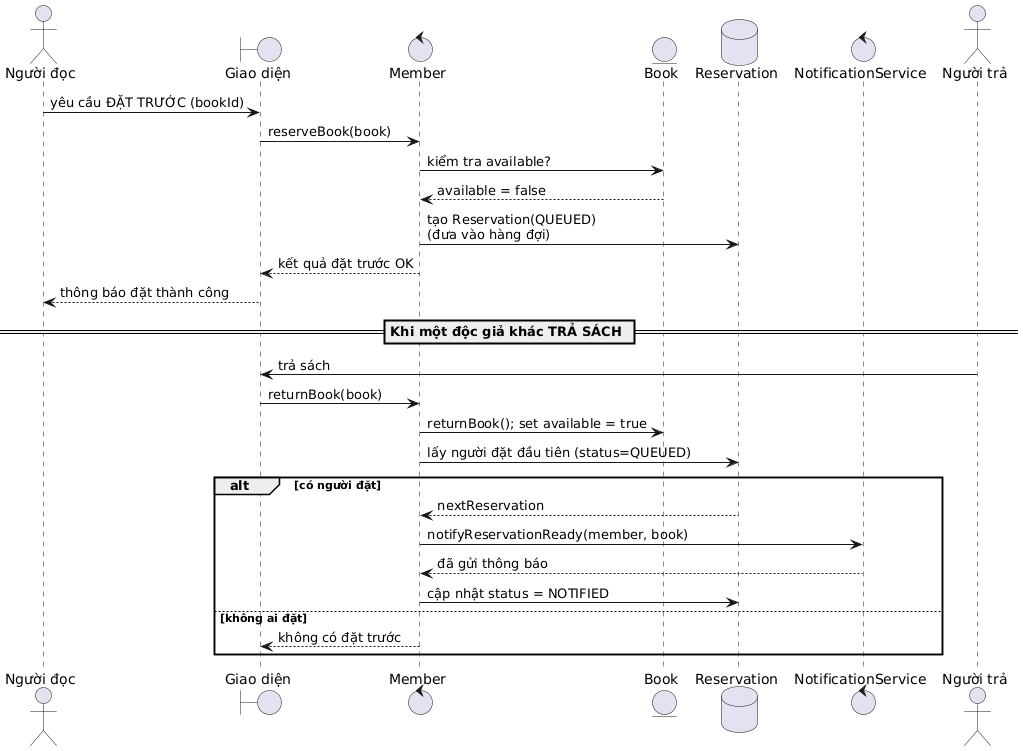
## Sơ đồ

Use case

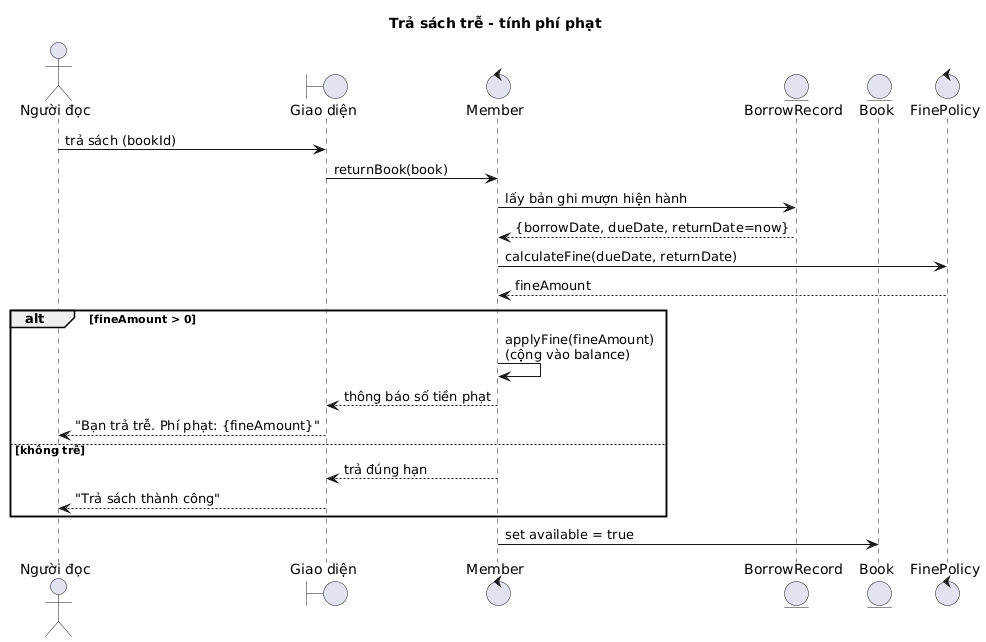


Class  


Sequence – Đặt trước & Thông báo khi trả



Sequence Diagram – Trả sách trễ & tính phí phạt



## Script PlantUML

Use case

@startuml

left to right direction

skinparam actorStyle awesome

skinparam packageStyle rectangle

rectangle "Hệ thống Quản lý Thư viện" as System {

usecase "Mượn sách" as UC\_Borrow

usecase "Trả sách" as UC\_Return

usecase "Xem lịch sử mượn" as UC\_History

usecase "Đặt sách trước" as UC\_Reserve

usecase "Thông báo khi sách sẵn sàng\n(cho người đặt đầu tiên)" as UC\_Notify

usecase "Đăng nhập" as UC\_Login

usecase "Quản lý sách" as UC\_BookMgr

usecase "Tạo tài khoản" as UC\_CreateAcc

usecase "Khóa tài khoản" as UC\_LockAcc

}

actor "Người đọc" as Reader

actor "Thủ thư" as Librarian

actor "Quản trị viên hệ thống" as Admin

Reader --> UC\_Borrow

Reader --> UC\_Return

Reader --> UC\_History

Reader --> UC\_Reserve

Reader --> UC\_Login

Librarian --> UC\_Login

Librarian --> UC\_BookMgr

Admin --> UC\_Login

Admin --> UC\_CreateAcc

Admin --> UC\_LockAcc

' Khi có đơn đặt trước, thao tác Trả sách sẽ kích hoạt thông báo

UC\_Notify ..> UC\_Reserve : <<include>>

UC\_Notify ..> UC\_Return : <<extend>>\n[điều kiện: có đặt trước]

@enduml

Class  
@startuml

skinparam classAttributeIconSize 0

' ====== Lớp người dùng & kế thừa ======

class User {

- userId : int

- name : String

- email : String

- status : UserStatus

+ login() : void

+ logout() : void

}

enum UserStatus {

ACTIVE

LOCKED

}

class Member {

- balance : decimal

+ borrowBook(book : Book) : void

+ returnBook(book : Book) : void

+ reserveBook(book : Book) : void

+ applyFine(amount : decimal) : void

}

class Librarian {

- staffId : int

+ addBook(book : Book) : void

+ removeBook(book : Book) : void

}

class Admin {

+ createAccount(user : User) : void

+ lockAccount(user : User) : void

}

User <|-- Member

User <|-- Librarian

User <|-- Admin

' ====== Miền nghiệp vụ sách ======

class Book {

- id : int

- title : String

- author : String

- available : boolean

+ borrow() : void

+ returnBook() : void

}

class BorrowRecord {

- recordId : int

- borrowDate : Date

- dueDate : Date

- returnDate : Date

}

class Reservation {

- reservationId : int

- reservedAt : Date

- status : ReservationStatus

}

enum ReservationStatus {

QUEUED

NOTIFIED

CANCELLED

FULFILLED

}

class NotificationService {

+ notifyReservationReady(m : Member, b : Book) : void

}

class FinePolicy {

- dailyRate : decimal

+ calculateFine(due : Date, ret : Date) : decimal

}

' ====== Quan hệ ======

Librarian "1" -- "0..\*" Book : manages >

Member "1" -- "0..\*" BorrowRecord

Book "1" -- "0..\*" BorrowRecord

Member "1" -- "0..\*" Reservation

Book "1" -- "0..\*" Reservation

Member ..> FinePolicy : uses >

Member ..> NotificationService : uses >

@enduml

Sequence – Đặt trước & Thông báo khi trả

@startuml

title Đặt sách trước & thông báo khi sách sẵn sàng

actor "Người đọc" as Reader

boundary "Giao diện" as UI

control "Member" as M

entity "Book" as B

database "Reservation" as R

control "NotificationService" as N

' --- Pha 1: Người đọc đặt trước khi sách đang mượn ---

Reader -> UI : yêu cầu ĐẶT TRƯỚC (bookId)

UI -> M : reserveBook(book)

M -> B : kiểm tra available?

B --> M : available = false

M -> R : tạo Reservation(QUEUED)\n(đưa vào hàng đợi)

M --> UI : kết quả đặt trước OK

UI --> Reader : thông báo đặt thành công

== Khi một độc giả khác TRẢ SÁCH ==

actor "Người trả" as Returner

Returner -> UI : trả sách

UI -> M : returnBook(book)

M -> B : returnBook(); set available = true

M -> R : lấy người đặt đầu tiên (status=QUEUED)

alt có người đặt

R --> M : nextReservation

M -> N : notifyReservationReady(member, book)

N --> M : đã gửi thông báo

M -> R : cập nhật status = NOTIFIED

else không ai đặt

M --> UI : không có đặt trước

end

@enduml

Sequence Diagram – Trả sách trễ & tính phí phạt

@startuml

actor "Người đọc" as Reader

boundary "Giao diện" as UI

control "Member" as M

entity "BorrowRecord" as BR

entity "Book" as B

control "FinePolicy" as FP

Reader -> UI : trả sách (bookId)

UI -> M : returnBook(book)

M -> BR : lấy bản ghi mượn hiện hành

BR --> M : {borrowDate, dueDate, returnDate=now}

M -> FP : calculateFine(dueDate, returnDate)

FP --> M : fineAmount

alt fineAmount > 0

M -> M : applyFine(fineAmount)\n(cộng vào balance)

M --> UI : thông báo số tiền phạt

UI --> Reader : "Bạn trả trễ. Phí phạt: {fineAmount}"

else không trễ

M --> UI : trả đúng hạn

UI --> Reader : "Trả sách thành công"

end

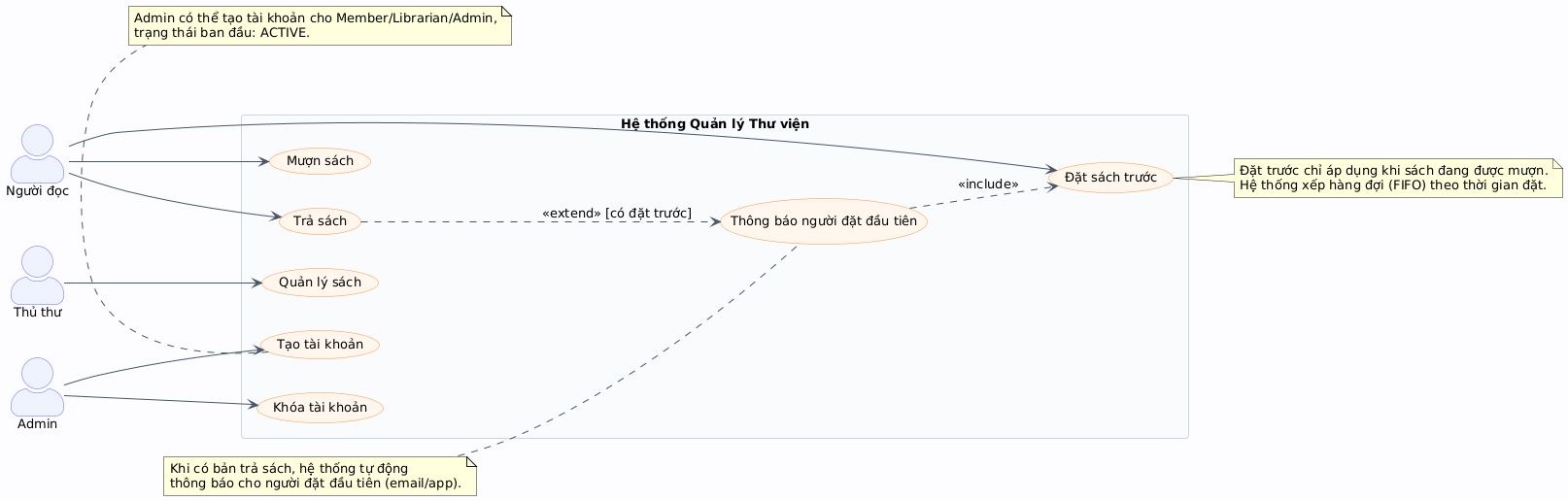
M -> B : set available = true

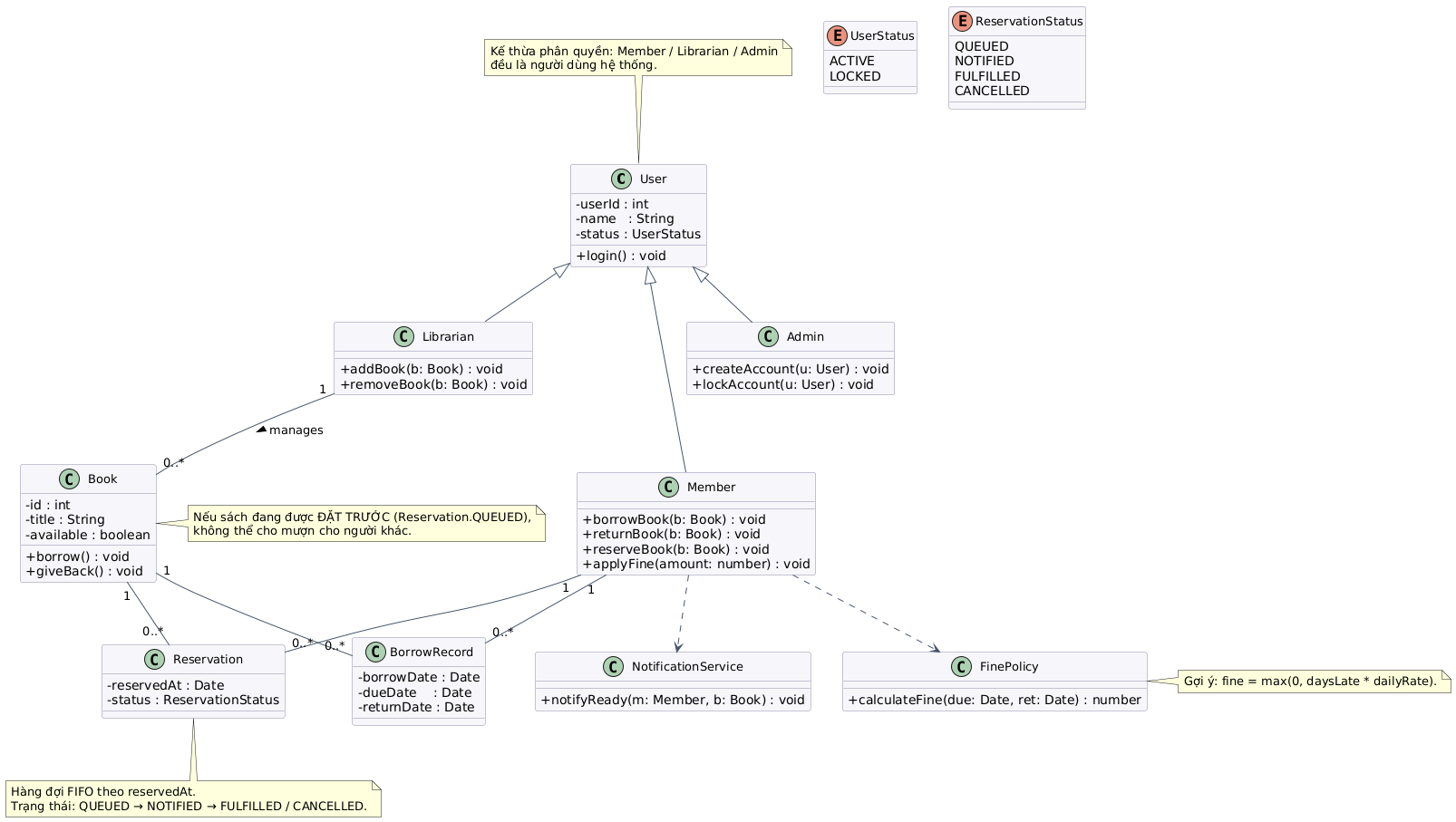
@enduml

# Task 5. Bổ sung tùy biến

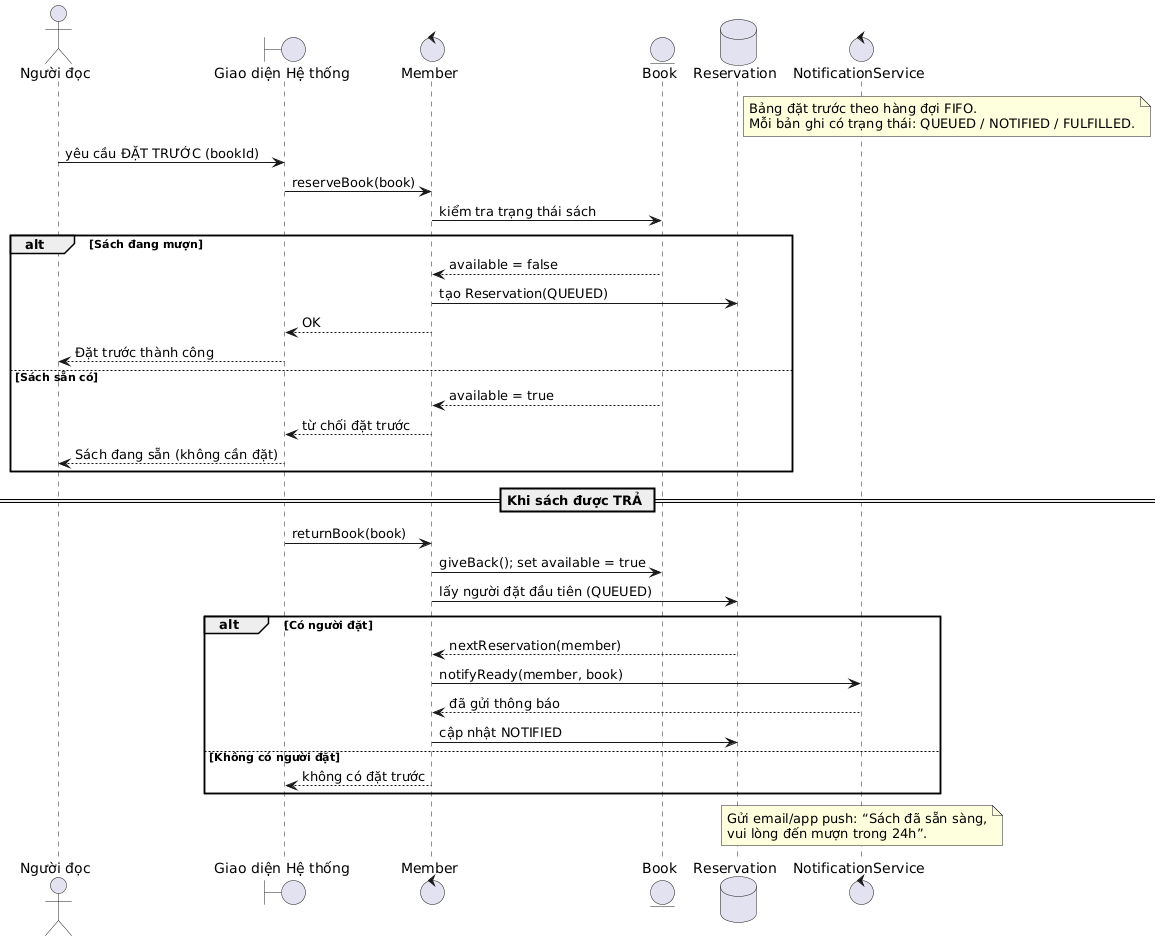
## Sơ đồ

Use case

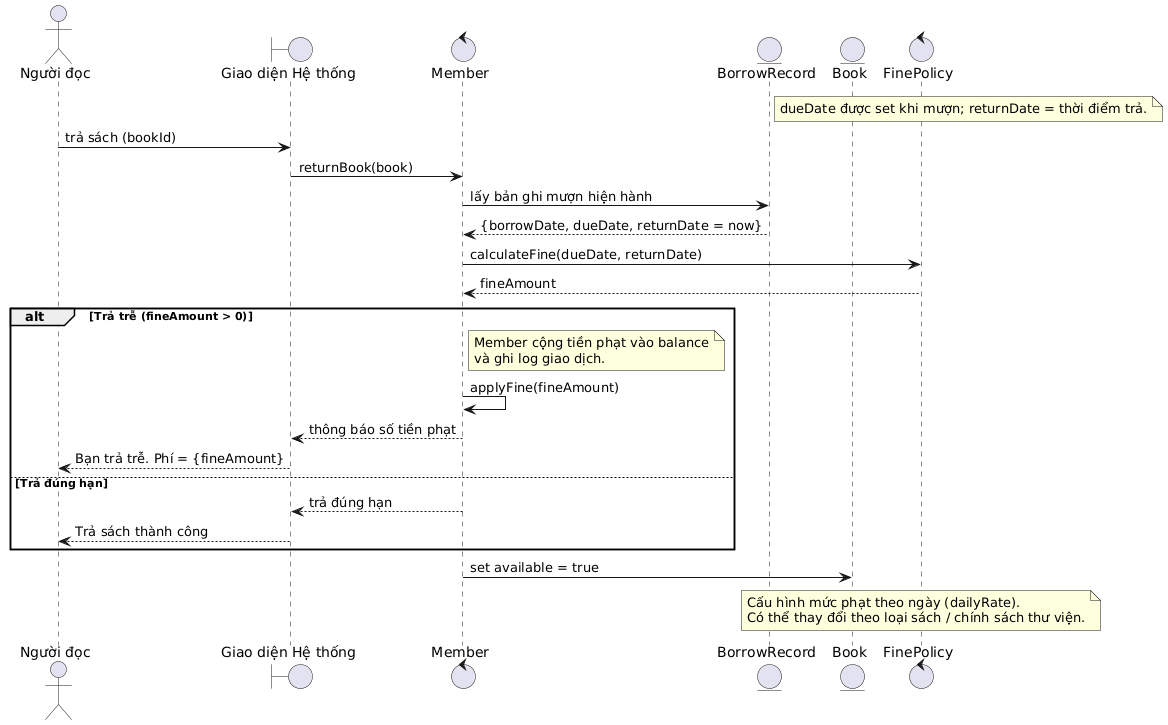


Class  


Sequence – Đặt trước & Thông báo khi trả



Sequence Diagram – Trả sách trễ & tính phí phạt



## Script PlantUML

Use case

@startuml

' ===== Style =====

left to right direction

skinparam shadowing false

skinparam backgroundColor #FCFCFF

skinparam defaultFontName Arial

skinparam defaultFontSize 13

skinparam ArrowColor #475569

skinparam packageStyle rectangle

skinparam packageBorderColor #CBD5E1

skinparam packageBackgroundColor #F8FAFC

skinparam actorStyle awesome

skinparam Actor {

BackgroundColor #EEF2FF

BorderColor #6366F1

}

skinparam UseCase {

BackgroundColor #FFF7ED

BorderColor #FB923C

}

' ===== Model =====

actor "Người đọc" as Reader

actor "Thủ thư" as Librarian

actor "Admin" as Admin

rectangle "Hệ thống Quản lý Thư viện" as System {

usecase "Mượn sách" as UC\_Borrow

usecase "Trả sách" as UC\_Return

usecase "Đặt sách trước" as UC\_Reserve

usecase "Thông báo người đặt đầu tiên" as UC\_Notify

usecase "Quản lý sách" as UC\_BookMgr

usecase "Tạo tài khoản" as UC\_Create

usecase "Khóa tài khoản" as UC\_Lock

}

Reader --> UC\_Borrow

Reader --> UC\_Return

Reader --> UC\_Reserve

Librarian --> UC\_BookMgr

Admin --> UC\_Create

Admin --> UC\_Lock

' Luồng thông báo gắn với trả sách khi có đặt trước

UC\_Notify ..> UC\_Reserve : <<include>>

UC\_Return ..> UC\_Notify : <<extend>> [có đặt trước]

' ===== Notes =====

note right of UC\_Reserve

Đặt trước chỉ áp dụng khi sách đang được mượn.

Hệ thống xếp hàng đợi (FIFO) theo thời gian đặt.

end note

note left of UC\_Notify

Khi có bản trả sách, hệ thống tự động

thông báo cho người đặt đầu tiên (email/app).

end note

note bottom of UC\_Create

Admin có thể tạo tài khoản cho Member/Librarian/Admin,

trạng thái ban đầu: ACTIVE.

end note

@enduml

Class  
@startuml

' ===== Style =====

skinparam classAttributeIconSize 0

skinparam BackgroundColor #FFFFFF

skinparam Class {

BackgroundColor #F7F7FB

BorderColor #8A8AA8

FontSize 13

}

skinparam ArrowThickness 1

skinparam ArrowColor #475569

' ===== Model =====

class User {

- userId : int

- name : String

- status : UserStatus

+ login() : void

}

enum UserStatus {

ACTIVE

LOCKED

}

class Member {

+ borrowBook(b: Book) : void

+ returnBook(b: Book) : void

+ reserveBook(b: Book) : void

+ applyFine(amount: number) : void

}

class Librarian {

+ addBook(b: Book) : void

+ removeBook(b: Book) : void

}

class Admin {

+ createAccount(u: User) : void

+ lockAccount(u: User) : void

}

User <|-- Member

User <|-- Librarian

User <|-- Admin

class Book {

- id : int

- title : String

- available : boolean

+ borrow() : void

+ giveBack() : void

}

class BorrowRecord {

- borrowDate : Date

- dueDate : Date

- returnDate : Date

}

class Reservation {

- reservedAt : Date

- status : ReservationStatus

}

enum ReservationStatus {

QUEUED

NOTIFIED

FULFILLED

CANCELLED

}

class FinePolicy {

+ calculateFine(due: Date, ret: Date) : number

}

class NotificationService {

+ notifyReady(m: Member, b: Book) : void

}

' ===== Quan hệ =====

Librarian "1" -- "0..\*" Book : manages >

Member "1" -- "0..\*" BorrowRecord

Book "1" -- "0..\*" BorrowRecord

Member "1" -- "0..\*" Reservation

Book "1" -- "0..\*" Reservation

Member ..> FinePolicy

Member ..> NotificationService

' ===== Notes =====

note right of Book

Nếu sách đang được ĐẶT TRƯỚC (Reservation.QUEUED),

không thể cho mượn cho người khác.

end note

note bottom of Reservation

Hàng đợi FIFO theo reservedAt.

Trạng thái: QUEUED → NOTIFIED → FULFILLED / CANCELLED.

end note

note top of User

Kế thừa phân quyền: Member / Librarian / Admin

đều là người dùng hệ thống.

end note

note right of FinePolicy

Gợi ý: fine = max(0, daysLate \* dailyRate).

end note

@enduml

Sequence – Đặt trước & Thông báo khi trả

@startuml

' ===== Style =====

skinparam sequenceArrowThickness 1

skinparam participantPadding 30

skinparam BoxPadding 10

skinparam responseMessageBelowArrow true

skinparam lifelineStrategy solid

actor "Người đọc" as Reader

boundary "Giao diện Hệ thống" as UI

control "Member" as Member

entity "Book" as Book

database "Reservation" as R

control "NotificationService" as NS

note right of R

Bảng đặt trước theo hàng đợi FIFO.

Mỗi bản ghi có trạng thái: QUEUED / NOTIFIED / FULFILLED.

end note

Reader -> UI : yêu cầu ĐẶT TRƯỚC (bookId)

UI -> Member : reserveBook(book)

Member -> Book : kiểm tra trạng thái sách

alt Sách đang mượn

Book --> Member : available = false

Member -> R : tạo Reservation(QUEUED)

Member --> UI : OK

UI --> Reader : Đặt trước thành công

else Sách sẵn có

Book --> Member : available = true

Member --> UI : từ chối đặt trước

UI --> Reader : Sách đang sẵn (không cần đặt)

end

== Khi sách được TRẢ ==

UI -> Member : returnBook(book)

Member -> Book : giveBack(); set available = true

Member -> R : lấy người đặt đầu tiên (QUEUED)

alt Có người đặt

R --> Member : nextReservation(member)

Member -> NS : notifyReady(member, book)

NS --> Member : đã gửi thông báo

Member -> R : cập nhật NOTIFIED

else Không có người đặt

Member --> UI : không có đặt trước

end

note over NS

Gửi email/app push: “Sách đã sẵn sàng,

vui lòng đến mượn trong 24h”.

end note

@enduml

Sequence Diagram – Trả sách trễ & tính phí phạt

@startuml

' ===== Style =====

skinparam sequenceArrowThickness 1

skinparam participantPadding 30

skinparam BoxPadding 10

skinparam responseMessageBelowArrow true

actor "Người đọc" as Reader

boundary "Giao diện Hệ thống" as UI

control "Member" as Member

entity "BorrowRecord" as BR

entity "Book" as Book

control "FinePolicy" as FP

note right of BR

dueDate được set khi mượn; returnDate = thời điểm trả.

end note

Reader -> UI : trả sách (bookId)

UI -> Member : returnBook(book)

Member -> BR : lấy bản ghi mượn hiện hành

BR --> Member : {borrowDate, dueDate, returnDate = now}

Member -> FP : calculateFine(dueDate, returnDate)

FP --> Member : fineAmount

alt Trả trễ (fineAmount > 0)

note right of Member

Member cộng tiền phạt vào balance

và ghi log giao dịch.

end note

Member -> Member : applyFine(fineAmount)

Member --> UI : thông báo số tiền phạt

UI --> Reader : Bạn trả trễ. Phí = {fineAmount}

else Trả đúng hạn

Member --> UI : trả đúng hạn

UI --> Reader : Trả sách thành công

end

Member -> Book : set available = true

note over FP

Cấu hình mức phạt theo ngày (dailyRate).

Có thể thay đổi theo loại sách / chính sách thư viện.

end note

@enduml