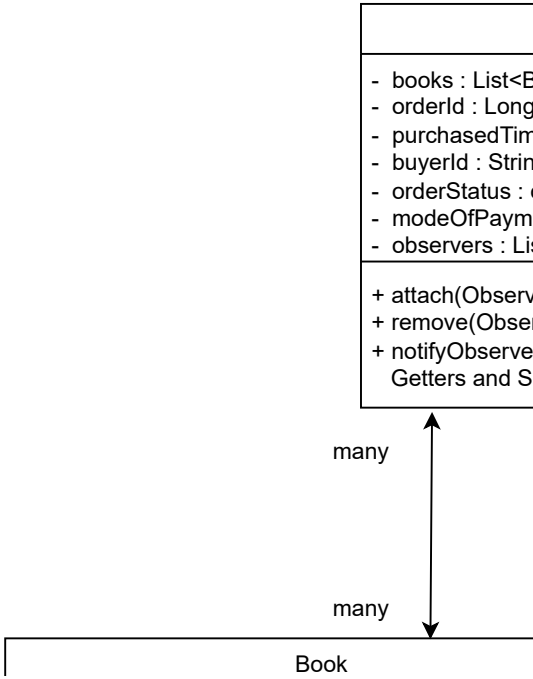
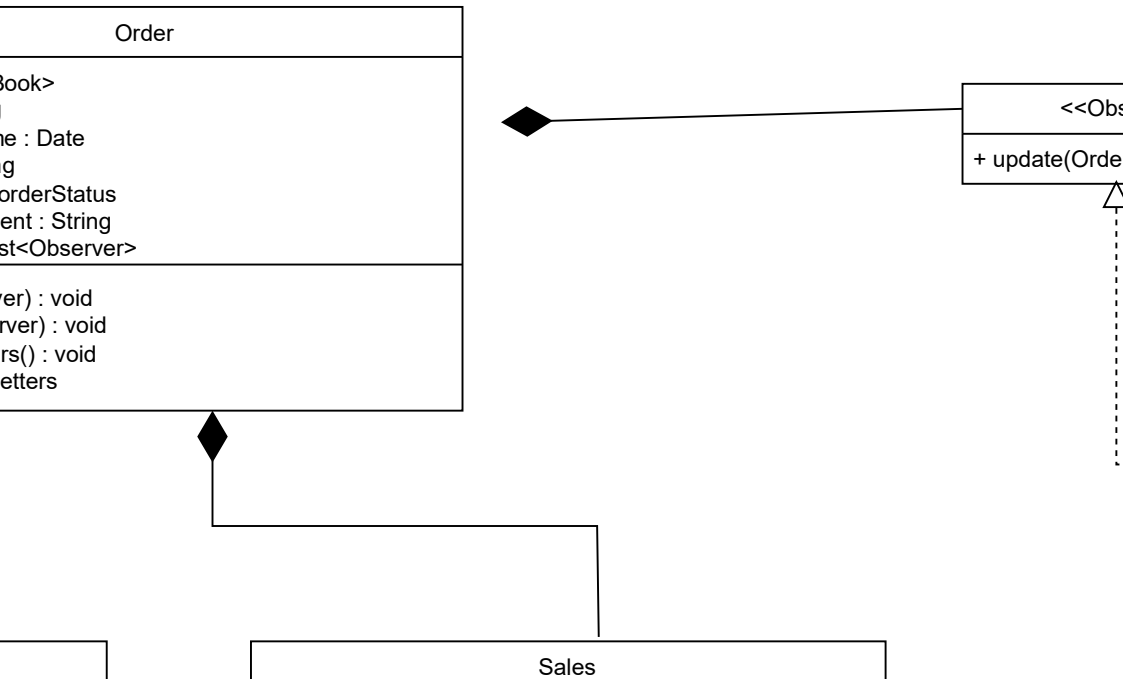
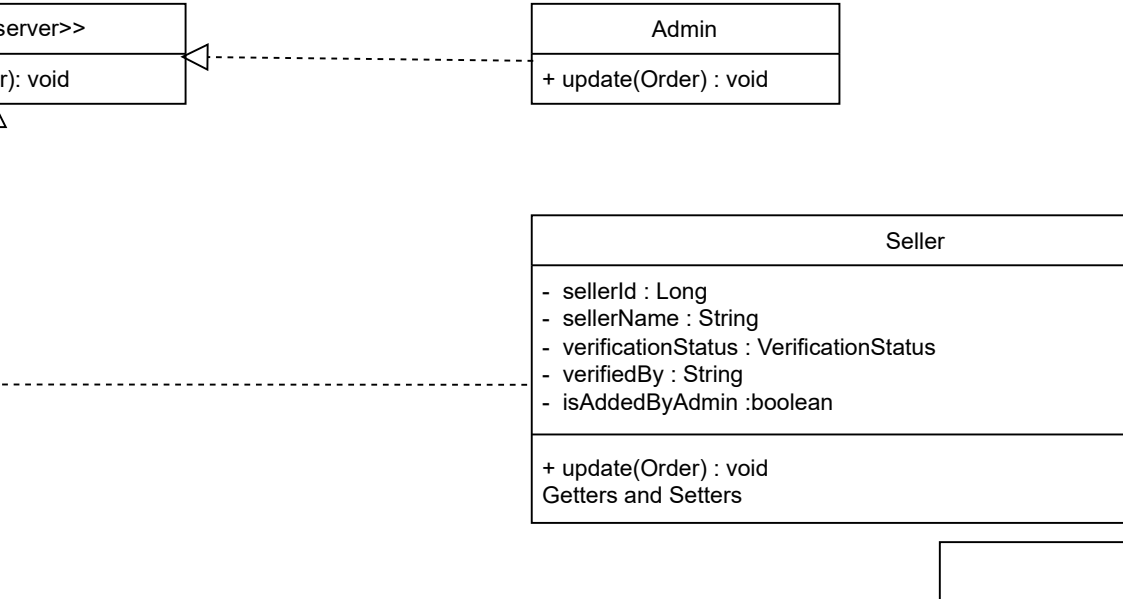
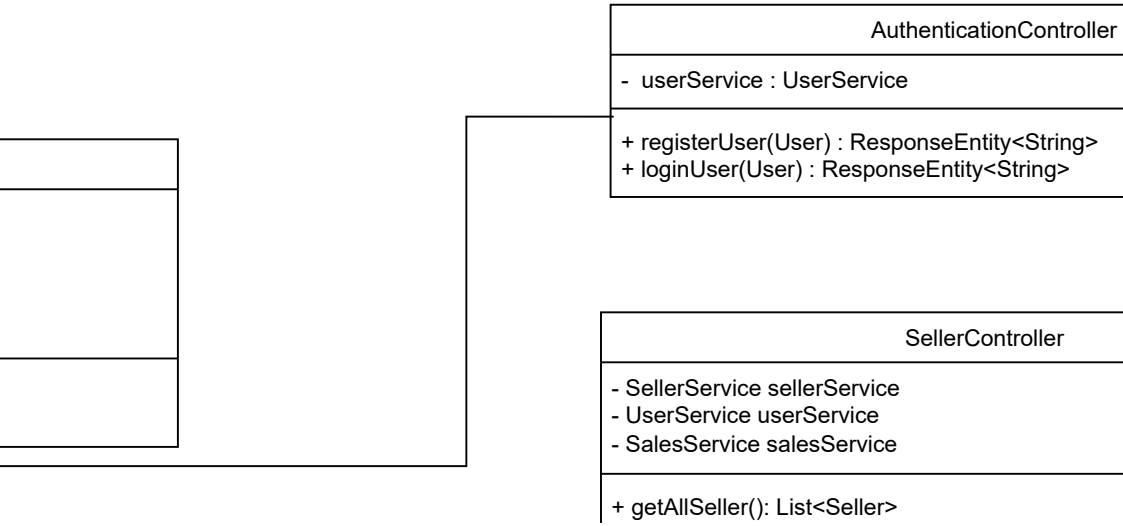
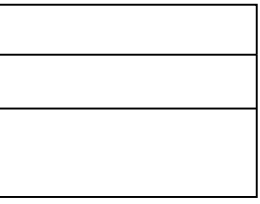
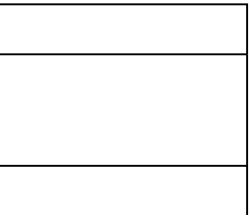

- books : List<B
- orderId : Long
- purchasedTim
- buyerId : Strin
- orderStatus :
- modeOfPaym
- observers : Lis
+ attach(Observ
+ remove(Obse
+ notifyObserve
Getters and S
many
many
Book

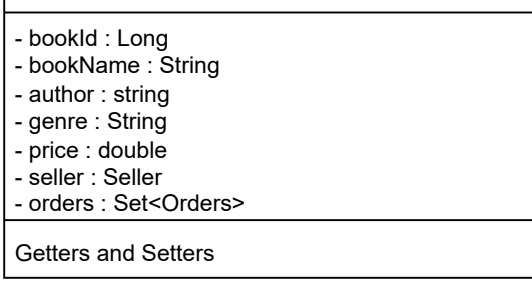

Order
Book>
ne : Date
g
orderStatus
ent : String
st<Observer>
ver) : void
rver) : void
rs() : void
etters
<<Obs
+ update(Orde
Sales

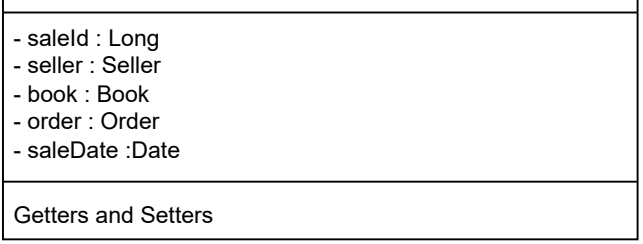

server>>
r): void
Admin
+ update(Order) : void
Seller
- sellerId : Long
- sellerName : String
- verificationStatus : VerificationStatus
- verifiedBy : String
- isAddedByAdmin :boolean
+ update(Order) : void
Getters and Setters

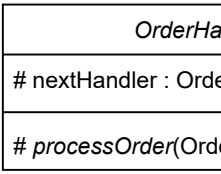

AuthenticationController
- userService : UserService
+ registerUser(User) : ResponseEntity<String>
+ loginUser(User) : ResponseEntity<String>
SellerController
- SellerService sellerService
- UserService userService
- SalesService salesService
+ getAllSeller(): List<Seller>

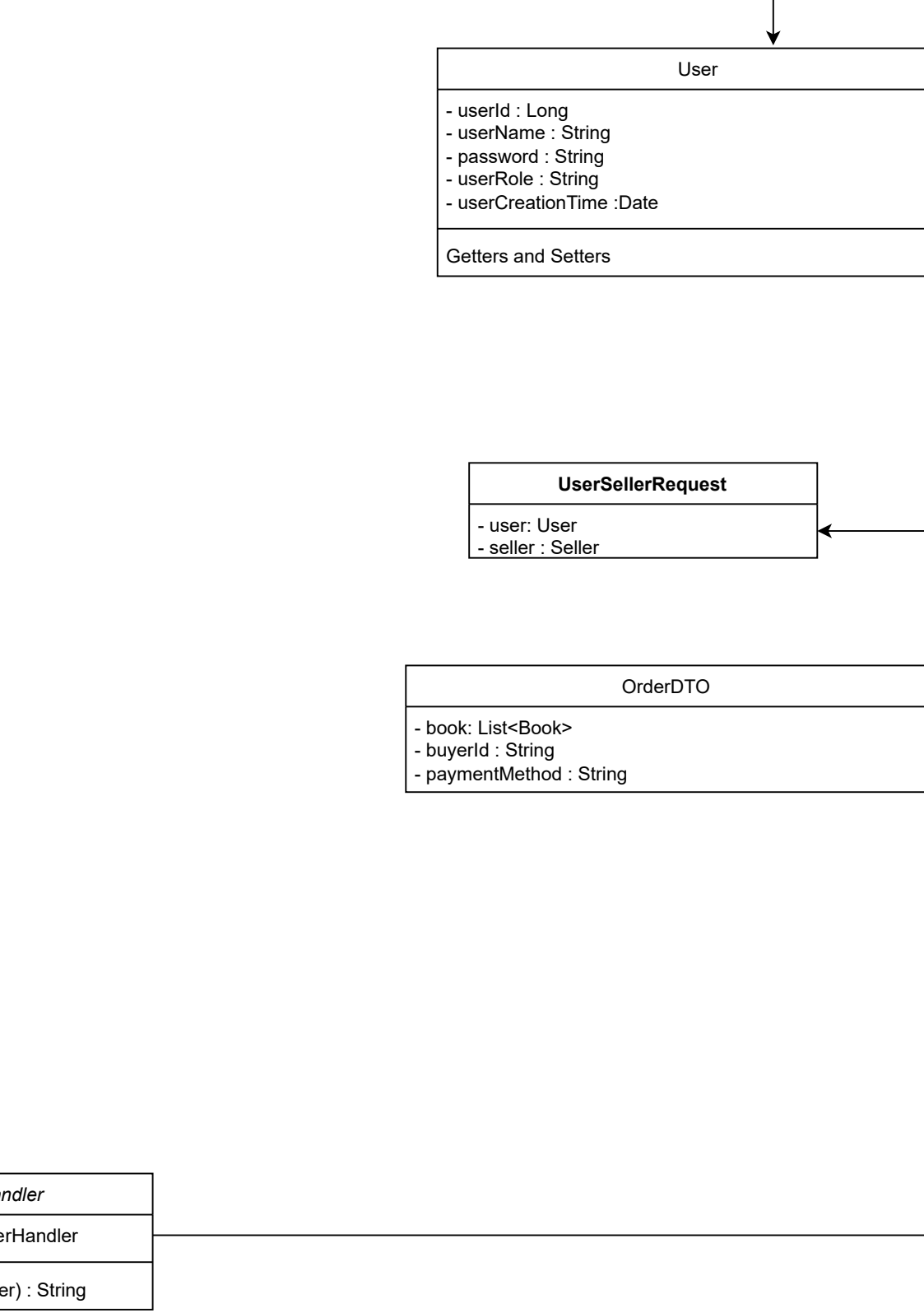


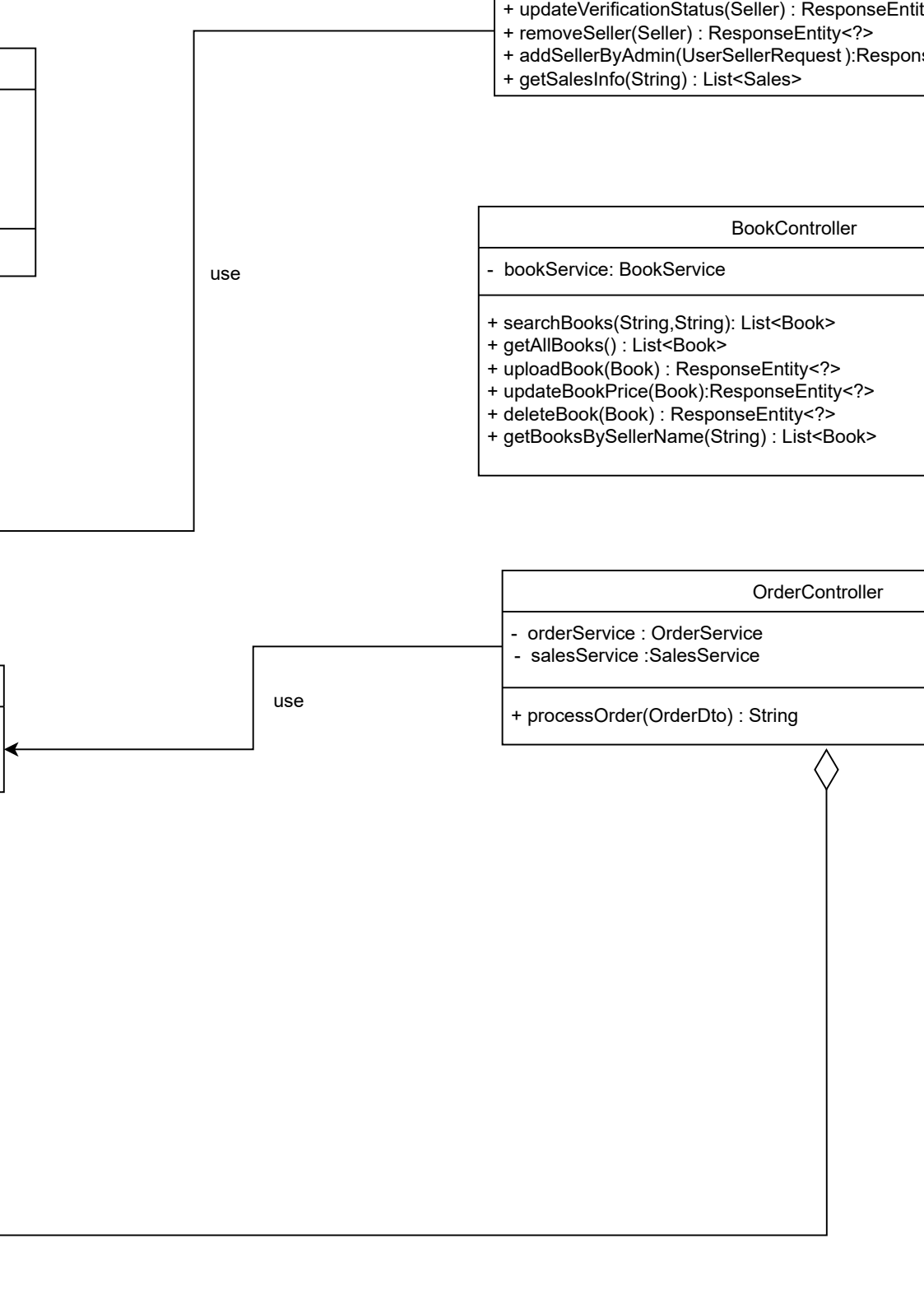


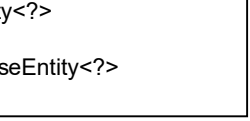

- bookId : Long
- bookName : String
- author : string
- genre : String
- price : double
- seller : Seller
- orders : Set<Orders>
Getters and Setters

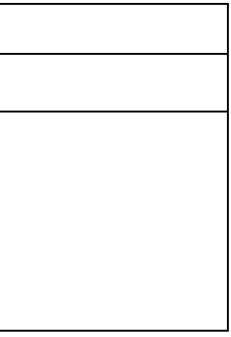

- saleId : Long
- seller : Seller
- book : Book
- order : Order
- saleDate :Date
Getters and Setters

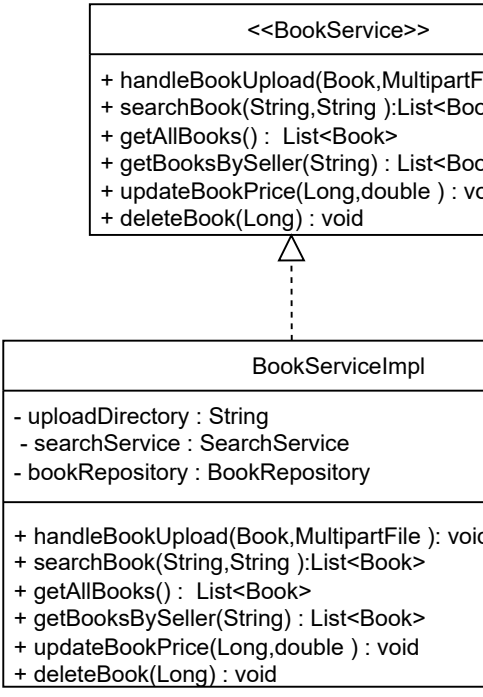

OrderHa
nextHandler : Orde
processOrder(Orde

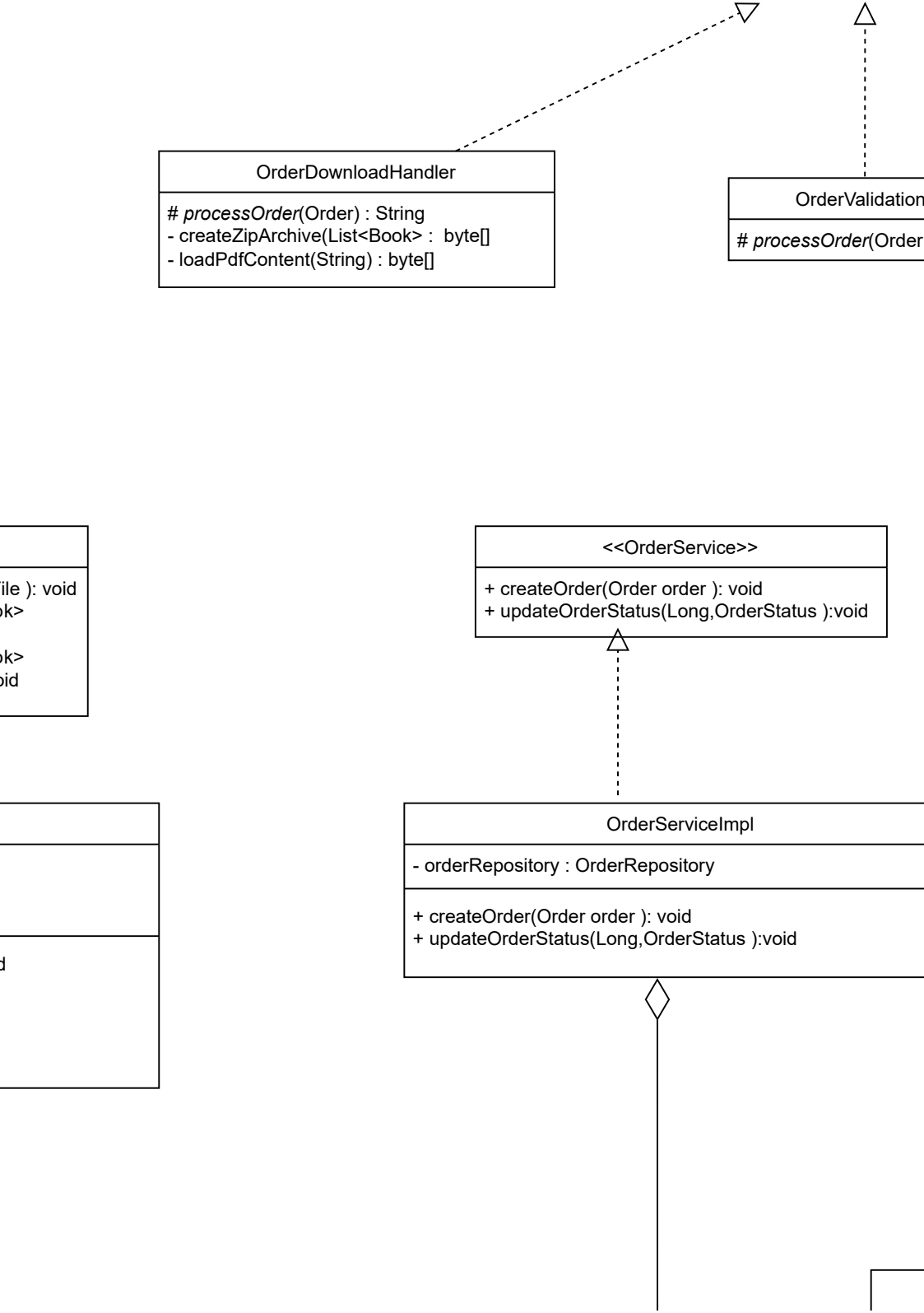

User
- userId : Long
- userName : String
- password : String
- userRole : String
- userCreationTime :Date
Getters and Setters
UserSellerRequest
- user: User
- seller : Seller
OrderDTO
- book: List<Book>
- buyerId : String
- paymentMethod : String
ndler
erHandler
er) : String

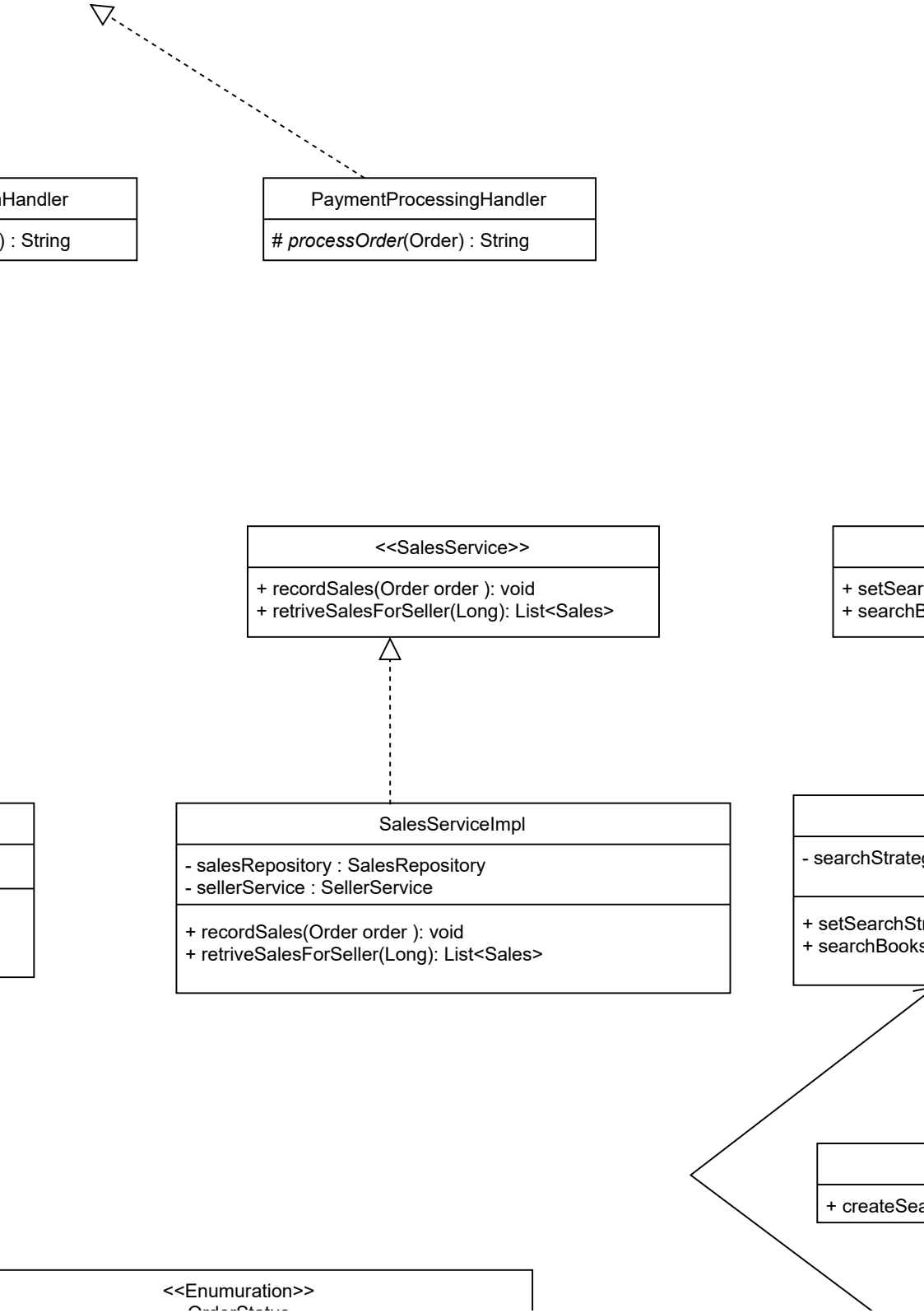

+ updateVerificationStatus(Seller) : ResponseEntit
+ removeSeller(Seller) : ResponseEntity<?>
+ addSellerByAdmin(UserSellerRequest):Respons
+ getSalesInfo(String) : List<Sales>
use
BookController
- bookService: BookService
+ searchBooks(String,String): List<Book>
+ getAllBooks() : List<Book>
+ uploadBook(Book) : ResponseEntity<?>
+ updateBookPrice(Book):ResponseEntity<?>
+ deleteBook(Book) : ResponseEntity<?>
+ getBooksBySellerName(String) : List<Book>
OrderController
- orderService : OrderService
- salesService :SalesService
+ processOrder(OrderDto) : String
use

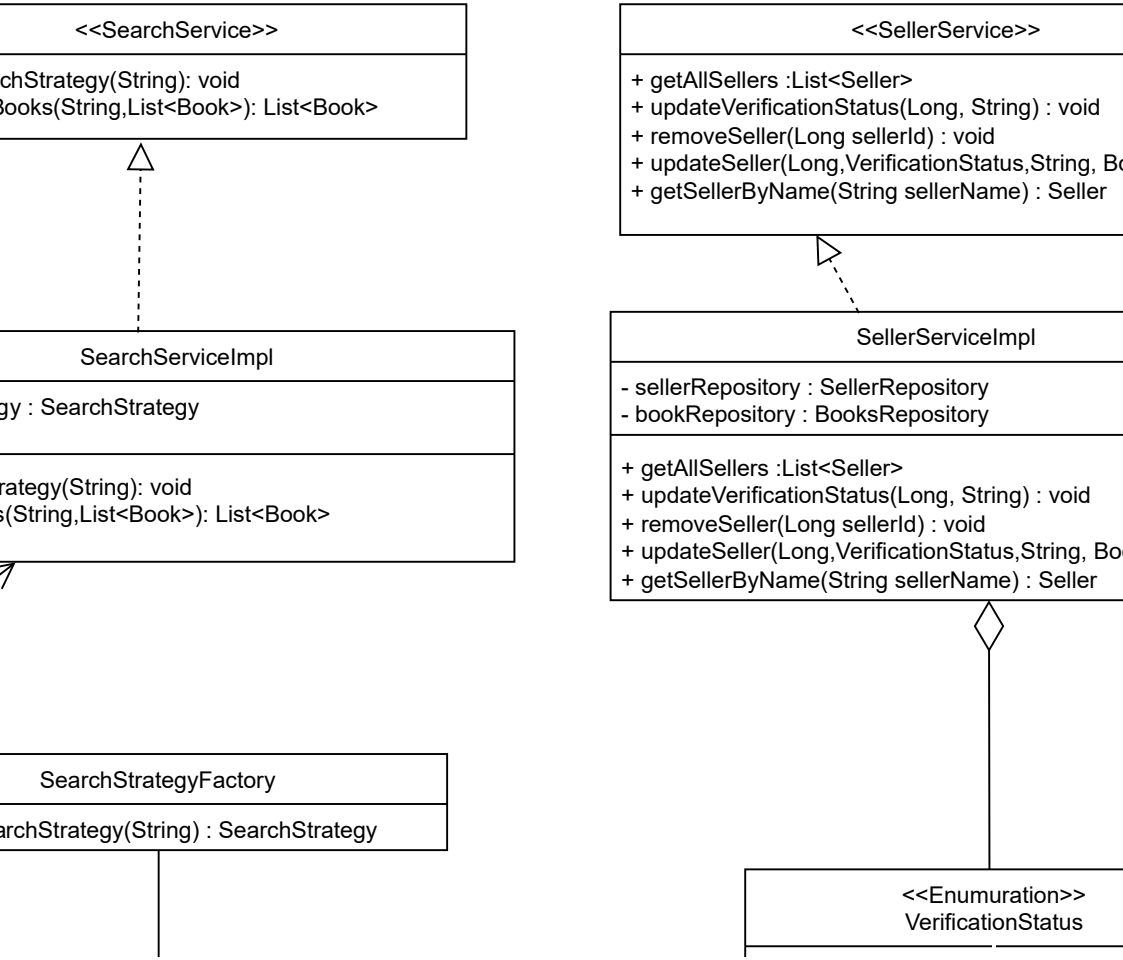

y<?>
seEntity<?>

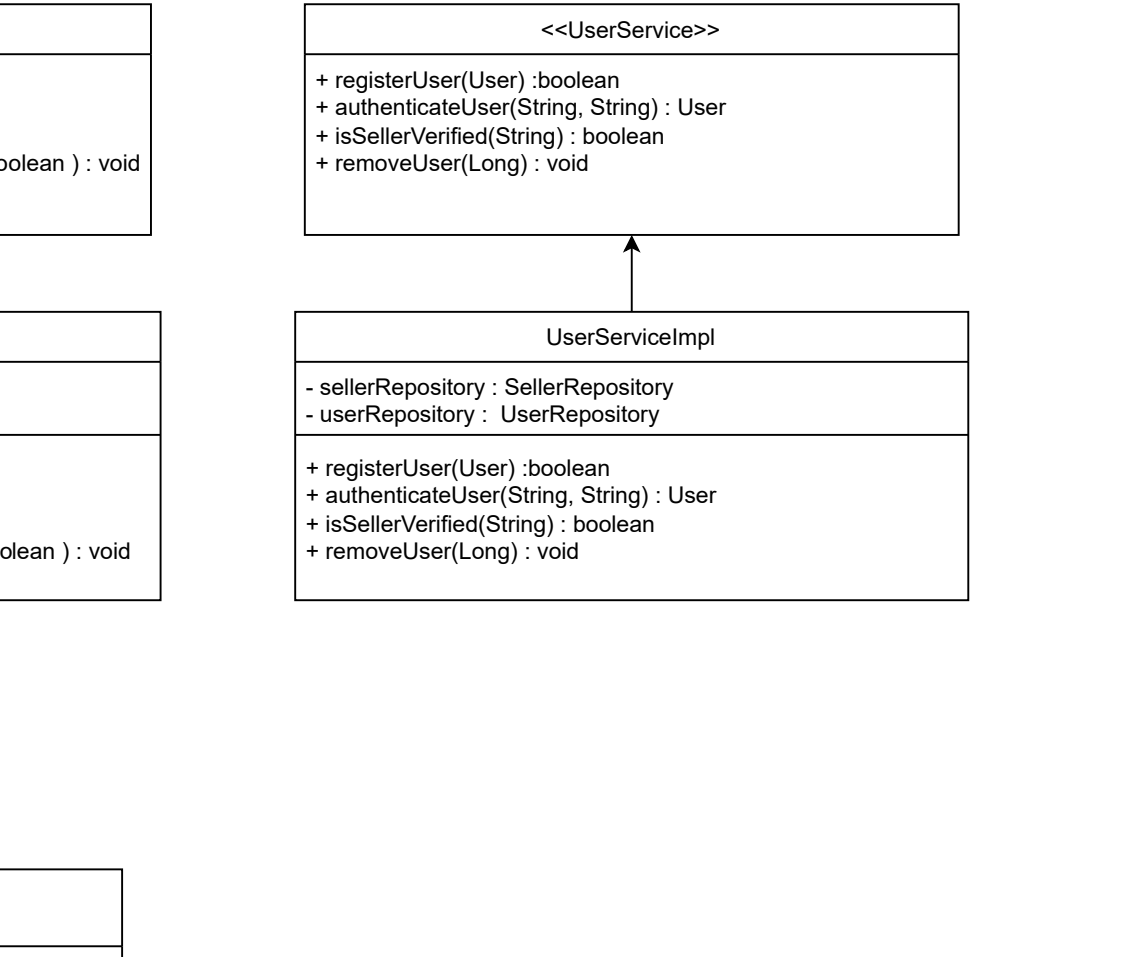


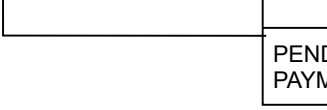

<<BookService>>
+ handleBookUpload(Book,MultipartF
+ searchBook(String,String):List<Boo
+ getAllBooks() : List<Book>
+ getBooksBySeller(String) : List<Boo
+ updateBookPrice(Long,double) : vo
+ deleteBook(Long) : void
BookServiceImpl
- uploadDirectory : String
- searchService : SearchService
- bookRepository : BookRepository
+ handleBookUpload(Book,MultipartFile): void
+ searchBook(String,String):List<Book>
+ getAllBooks() : List<Book>
+ getBooksBySeller(String) : List<Book>
+ updateBookPrice(Long,double) : void
+ deleteBook(Long) : void

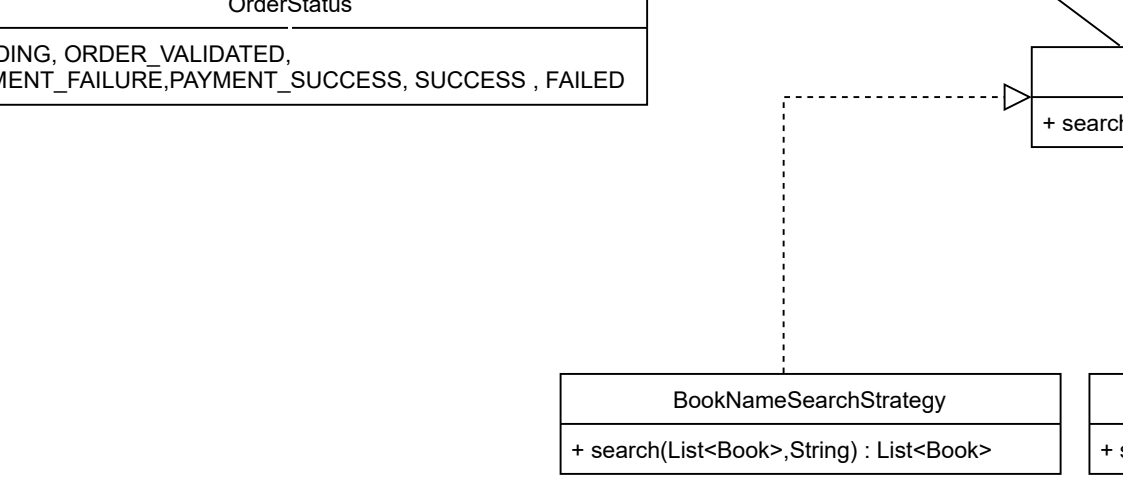

OrderDownloadHandler
processOrder(Order) : String
- createZipArchive(List<Book> : byte[]
- loadPdfContent(String) : byte[]
OrderValidation
processOrder(Order
<<OrderService>>
+ createOrder(Order order): void
+ updateOrderStatus(Long,OrderStatus):void
OrderServiceImpl
- orderRepository : OrderRepository
+ createOrder(Order order): void
+ updateOrderStatus(Long,OrderStatus):void

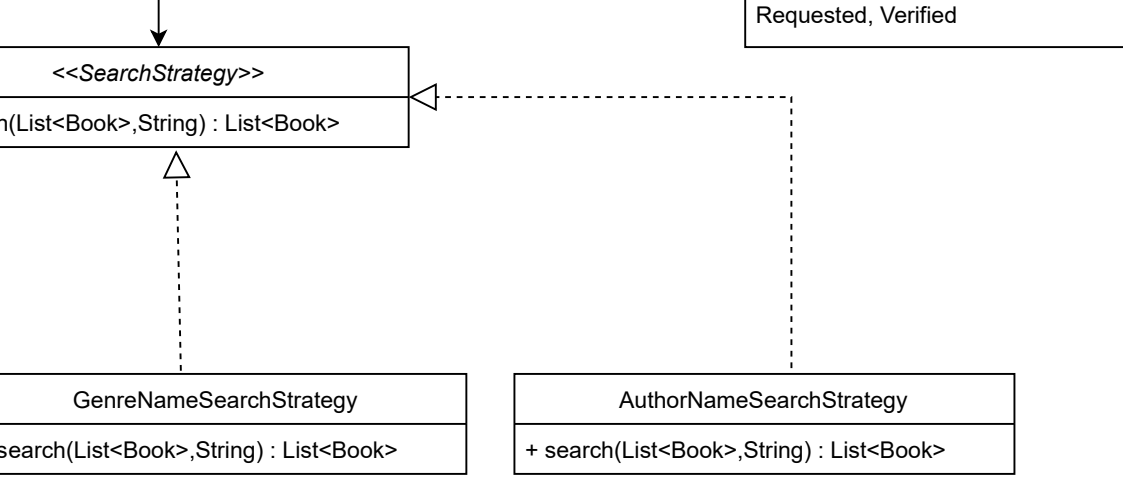

nHandler
) : String
PaymentProcessingHandler
processOrder(Order) : String
<<SalesService>>
+ recordSales(Order order): void
+ retriveSalesForSeller(Long): List<Sales>
+ setSear
+ searchB
SalesServiceImpl
- salesRepository : SalesRepository
- sellerService : SellerService
+ recordSales(Order order): void
+ retriveSalesForSeller(Long): List<Sales>
- searchStrateg
+ setSearchSt
+ searchBooks
+ createSea
<<Enumuration>>


<<SearchService>>
chStrategy(String): void
Books(String,List<Book>): List<Book>
SearchServiceImpl
gy : SearchStrategy
rategy(String): void
s(String,List<Book>): List<Book>
<<SellerService>>
+ getAllSellers :List<Seller>
+ updateVerificationStatus(Long, String) : void
+ removeSeller(Long sellerId) : void
+ updateSeller(Long,VerificationStatus,String, Bo
+ getSellerByName(String sellerName) : Seller
SellerServiceImpl
- sellerRepository : SellerRepository
- bookRepository : BooksRepository
+ getAllSellers :List<Seller>
+ updateVerificationStatus(Long, String) : void
+ removeSeller(Long sellerId) : void
+ updateSeller(Long,VerificationStatus,String, Bo
+ getSellerByName(String sellerName) : Seller
SearchStrategyFactory
archStrategy(String) : SearchStrategy
<<Enumuration>>
VerificationStatus


<<UserService>>
+ registerUser(User) :boolean
+ authenticateUser(String, String) : User
+ isSellerVerified(String) : boolean
+ removeUser(Long) : void
UserServiceImpl
- sellerRepository : SellerRepository
- userRepository : UserRepository
+ registerUser(User) :boolean
+ authenticateUser(String, String) : User
+ isSellerVerified(String) : boolean
+ removeUser(Long) : void


PEND
PAYM


OrderStatus
DING, ORDER_VALIDATED,
MENT_FAILURE,PAYMENT_SUCCESS, SUCCESS , FAILED
+ search
BookNameSearchStrategy
+ search(List<Book>,String) : List<Book>
+ s


Requested, Verified
<<SearchStrategy>>
n(List<Book>,String) : List<Book>
GenreNameSearchStrategy
search(List<Book>,String) : List<Book>
AuthorNameSearchStrategy
+ search(List<Book>,String) : List<Book>