**[Design a Library Management System](https://github.com/tssovi/grokking-the-object-oriented-design-interview/blob/master/object-oriented-design-case-studies/design-a-library-management-system.md" \l "design-a-library-management-system)**

We will focus on the following set of requirements while designing the Library Management System:

1. Any library member should be able to search books by their title, author, subject category as well by the publication date.
2. Each book will have a unique identification number and other details including a rack number which will help to physically locate the book.
3. There could be more than one copy of a book, and library members should be able to check-out and reserve any copy. We will call each copy of a book, a book item.
4. The system should be able to retrieve information like who took a particular book or what are the books checked-out by a specific library member.
5. There should be a maximum limit (5) on how many books a member can check-out.
6. There should be a maximum limit (10) on how many days a member can keep a book.
7. The system should be able to collect fines for books returned after the due date.
8. Members should be able to reserve books that are not currently available.
9. The system should be able to send notifications whenever the reserved books become available, as well as when the book is not returned within the due date.
10. Each book and member card will have a unique barcode. The system will be able to read barcodes from books and members’ library cards.

enum BookStatus {

AVAILABLE, CHECKED\_OUT, RESERVED

}

Enum BookReservationStatus {

WAITING, PENDING\_APPROVAL, RESERVED, CANCELED

}

enum AccountActivationStatus {

ACTIVE, CLOSED, BLOCKED

}

Class Member {

AccountType accountType;

User user;

List<BookItem> memberBooks;

BarCodeSystem userBarCode;

List<BookItem> getCheckedOutBooks() {

}

void removeBookItemFromBooks(BookItem bookItem) {

this.memberBooks.remove(BookItem);

}

BookingStatus checkOutOrReserve(BookItem) {

If(BookItem.BookReservationStatus == AVAILABLE) {

} else {

}

}

}

class User {

String name;

String email;

String phoneNum;

Address address;

DateTime creationDate;

AccountActivationStatus status;

}

Class Address {

String streetName;

String country;

String state;

String zip;

String county;

String apartmentNumber;

}

Class BookItem {

Book bookId;

BarCodeSystem uniqueBookIdentificationNum;

BookReservationStatus isAvailable;

}

Class Book {

String title;

String author;

String Subject;

String category;

String BookId;

DateTime publicationDate;

Long rackNo;

BookStatus isAvailable;

List<Member> reservationList;

List<BookItem> bookItems;

Void addReservation(MemberId){

this.reservationList(MemberId);

}

}

Class BookReservation {

TImeStamp checkoutDate;

TimeStamp returnDate;

Member studentId;

Libarian librarian;

TimeStamp dueDate;

BookItem bookItem;

String reservationId;

String BookReservation(BookItem, Member, Librarian, TimeStamp) {

}

}

Class Library {

List<BookItem> listOfBooks;

List<Member> members;

void manageReservation (BookReservation reservationId) {

if(reservationId.getCheckOutDate> System.currentTimeMillis()) {

} else {

reservationId.setReturnId ();

reservationId.getBookItemId().setIsAvailable();

reservationId.getMemberId.removeBookItem(BookItem);

}

}

void lookUpAndReserve(Book bookId) {

if(bookId.isAvailable) {

//makeReservation

} else {

bookId.addReservation(MemberId);

}

}

List<BookItem> getBooksByMember(Member memberId) {

}

Member getMemberByBookId(BookItem bookItem) {

}

}

Class NotificationService{

}

Class BarCodeSystem {

}