Design a Library Management System

We'll cover the following

A Library Management System is a software built to handle the primary housekeeping functions of a library. Libraries rely on library management systems to manage asset collections as well as relationships with their members. Library management systems help libraries keep track of the books and their checkouts, as well as members' subscriptions and profiles.

Grokking the Object Oriented Design Interview

Library management systems also involve maintaining the database for entering new books and recording books that have been borrowed with their respective due dates.

System Requirements

💡 ***Always clarify requirements at the beginning of the***

interview. Be sure to ask questions to find the exact scope of the system that the interviewer has in mind.

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.

Use case diagram

We have three main actors in our system:

- **Librarian:** Mainly responsible for adding and modifying books, book items, and users. The Librarian can also issue, reserve, and return book items.
- **Member:** All members can search the catalog, as well as check-out, reserve, renew, and return a book.
- **System:** Mainly responsible for sending notifications for overdue books, canceled reservations, etc.

Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

64% completed

Search Course

oriented-design-interview/7npMYmzm8DA)

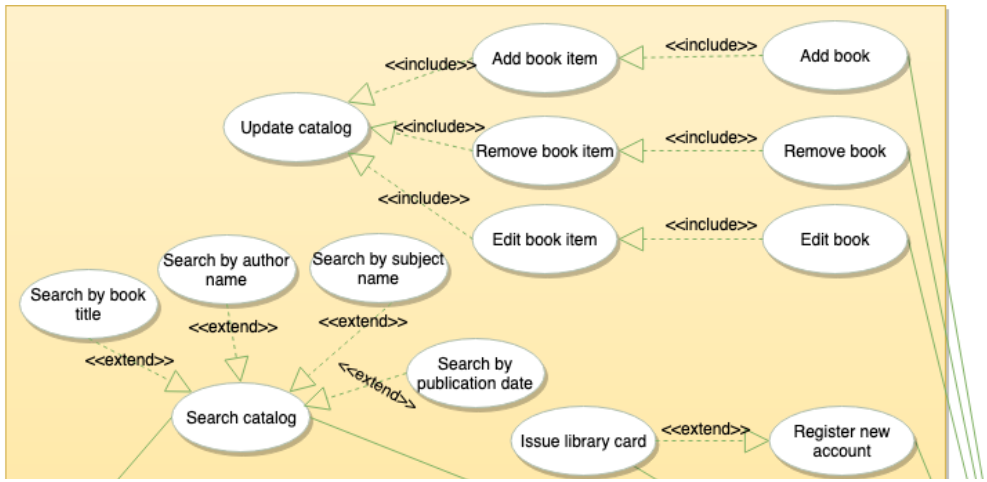
Class Diagram (/courses/grokking-the-object-oriented-design-interview/g7Lw3O0A2Aj)

Sequence diagram (/courses/grokking-the-object-oriented-design-interview/7nX38BMK9NO)

Activity Diagrams (/courses/grokking-the-object-oriented-design-interview/B8RPL3VEl8N)

Here are the top use cases of the Library Management System:

- **Add/Remove/Edit book:** To add, remove or modify a book or book item.
- **Search catalog:** To search books by title, author, subject or publication date.
- **Register new account/cancel membership:** To add a new member or cancel the membership of an existing member.
- **Check-out book:** To borrow a book from the library.
- **Reserve book:** To reserve a book which is not currently available.
- **Renew a book:** To reborrow an already checked-out book.
- **Return a book:** To return a book to the library which was issued to a member.



Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

64% completed

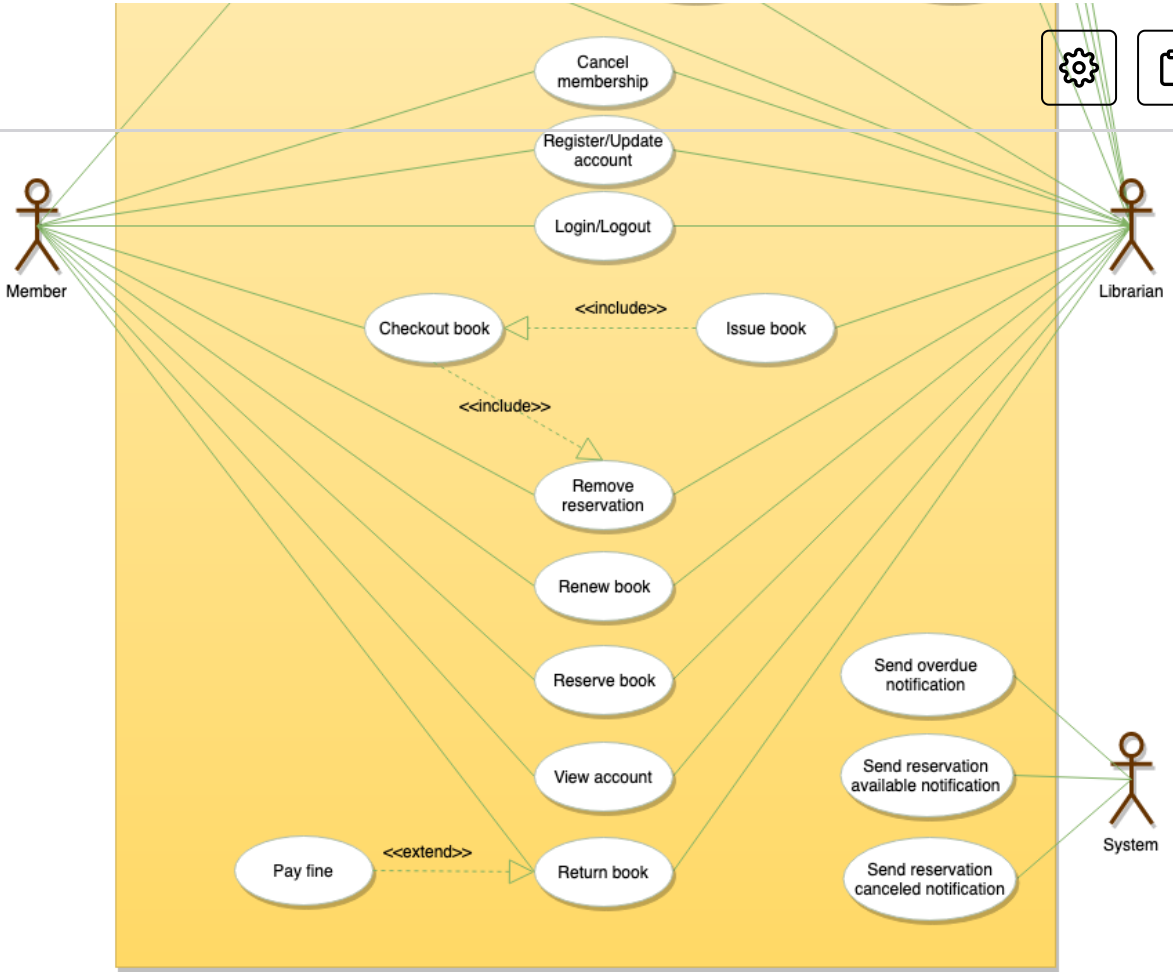
Search Course

oriented-design-interview/7npMYmzm8DA)

Class Diagram (/courses/grokking-the-object-oriented-design-interview/g7Lw3O0A2Aj)

Sequence diagram (/courses/grokking-the-object-oriented-design-interview/7nX38BMK9NO)

Activity Diagrams (/courses/grokking-the-object-oriented-design-interview/B8RPL3VEl8N)



Use case diagram

Class diagram

Here are the main classes of our Library Management System:

- **Library:** The central part of the organization for which this

software has been designed. It has attributes like 'Name' to distinguish it from any other libraries and 'Address' to describe its location.

- **Book:** The basic building block of the system. Every book will have ISBN, Title, Subject, Publishers, etc.
- **BookItem:** Any book can have multiple copies, each copy will be considered a book item in our system. Each book item will have a unique barcode.
- **Account:** We will have two types of accounts in the system, one will be a general member, and the other will be a librarian.
- **LibraryCard:** Each library user will be issued a library card, which will be used to identify users while issuing or returning books.
- **BookReservation:** Responsible for managing reservations against book items.
- **BookLending:** Manage the checking-out of book items.
- **Catalog:** Catalogs contain list of books sorted on certain criteria. Our system will support searching through four catalogs: Title, Author, Subject, and Publish-date.

Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

64% completed

Search Course

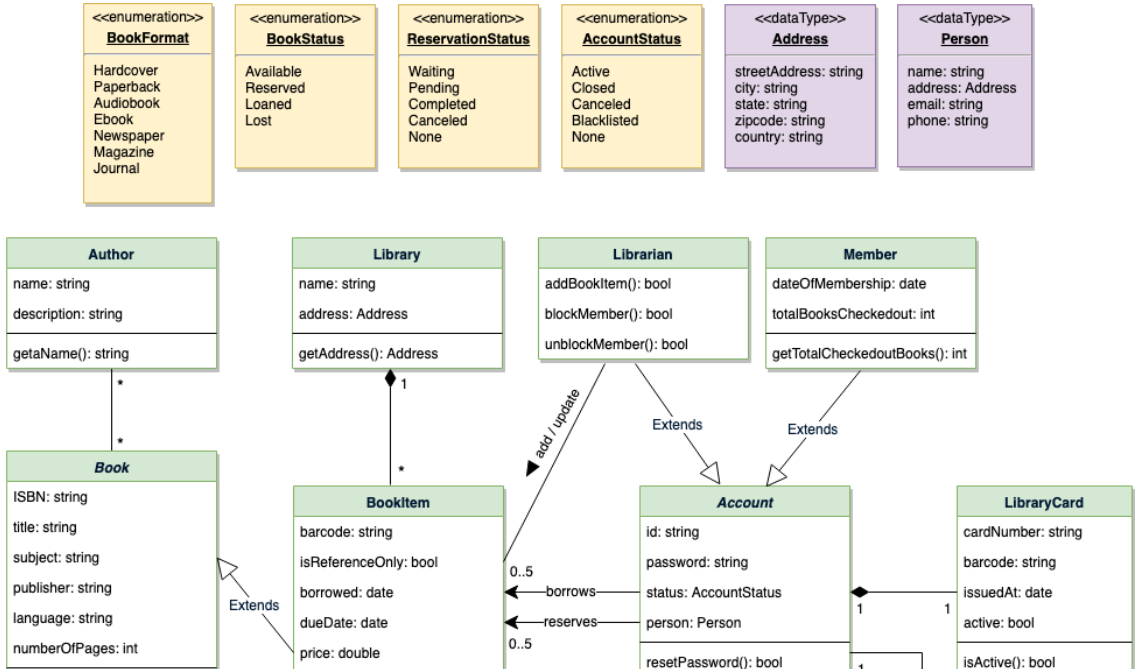
oriented-design-interview/7npMYmzm8DA)

Class Diagram (/courses/grokking-the-object-oriented-design-interview/g7Lw3O0A2Aj)

Sequence diagram (/courses/grokking-the-object-oriented-design-interview/7nX38BMK9NO)

Activity Diagrams (/courses/grokking-the-object-oriented-design-interview/B8RPL3VEl8N)

- **Fine:** This class will be responsible for calculating and collecting fines from library members.
- **Author:** This class will encapsulate a book author.
- **Rack:** Books will be placed on racks. Each rack will be identified by a rack number and will have a location identifier to describe the physical location of the rack in the library.
- **Notification:** This class will take care of sending notifications to library members.

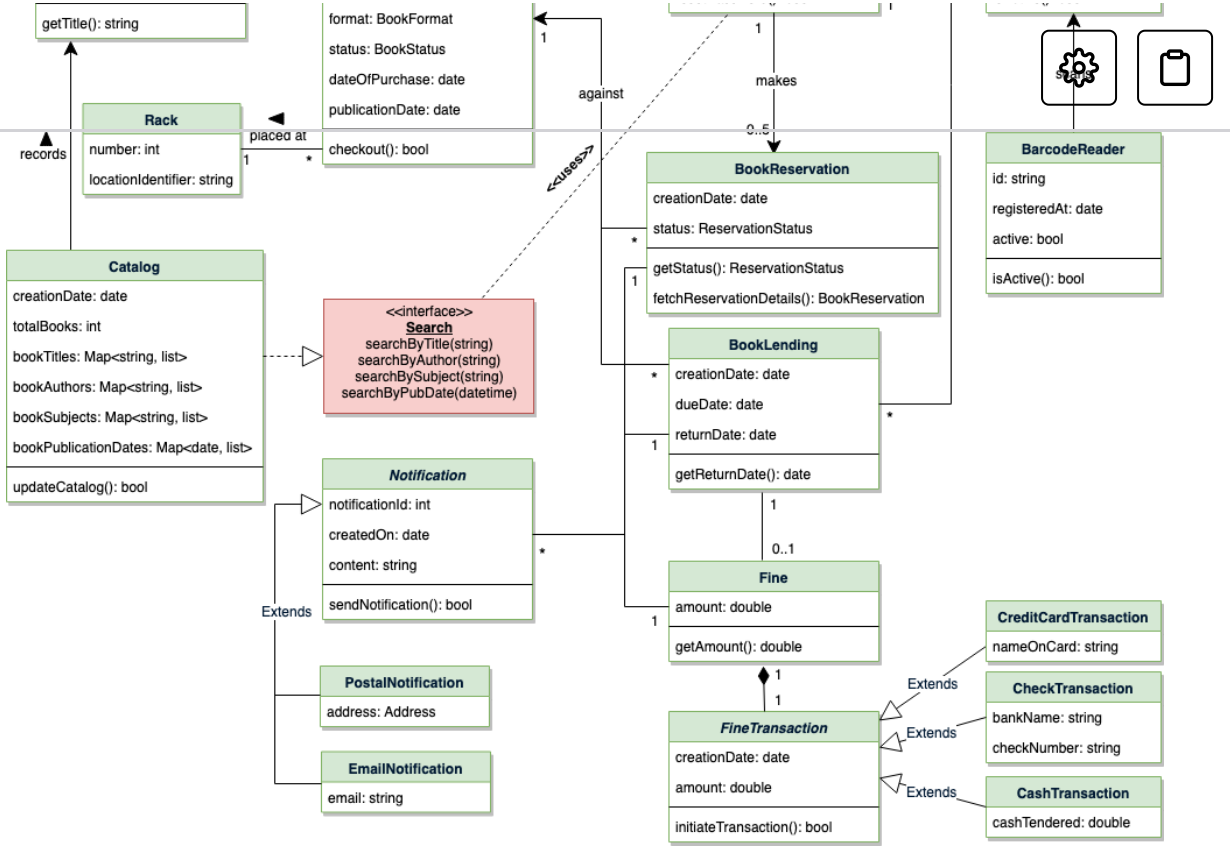


Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

64% completed

Search Course



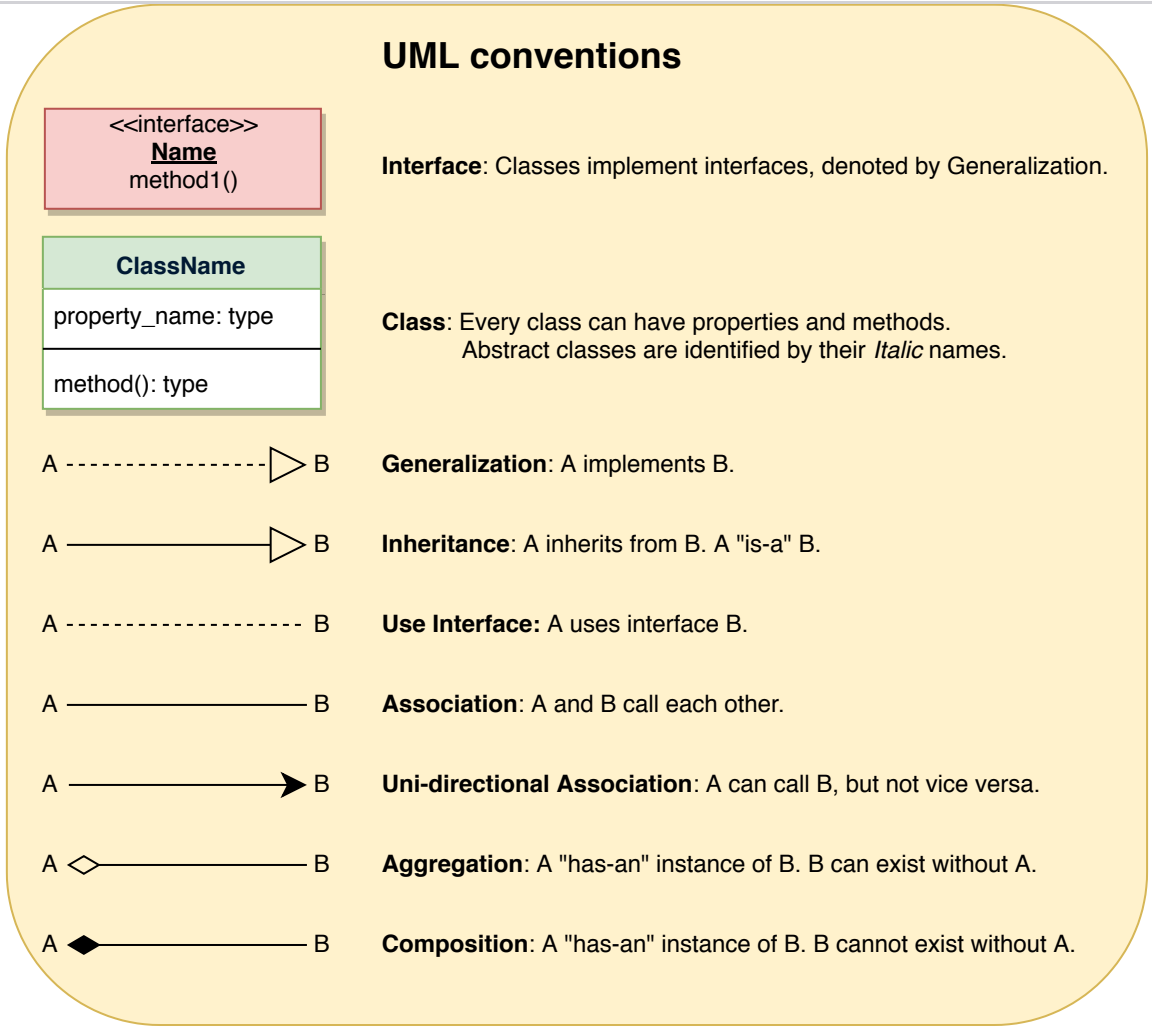
Class diagram for Library Management System

Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

64% completed

Search Course

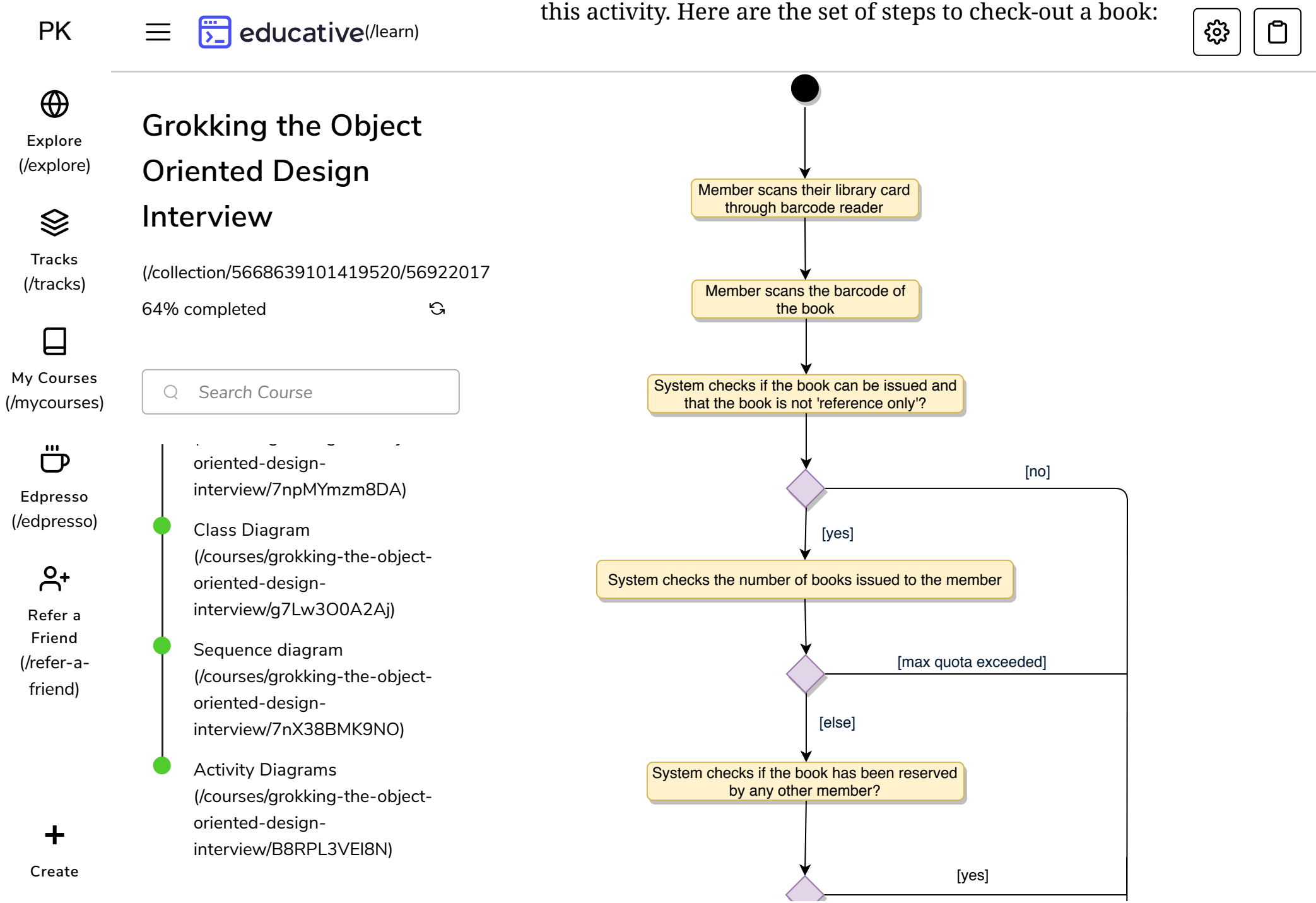


Activity diagrams

#

Check-out a book: Any library member or librarian can perform

this activity. Here are the set of steps to check-out a book:



Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

64% completed

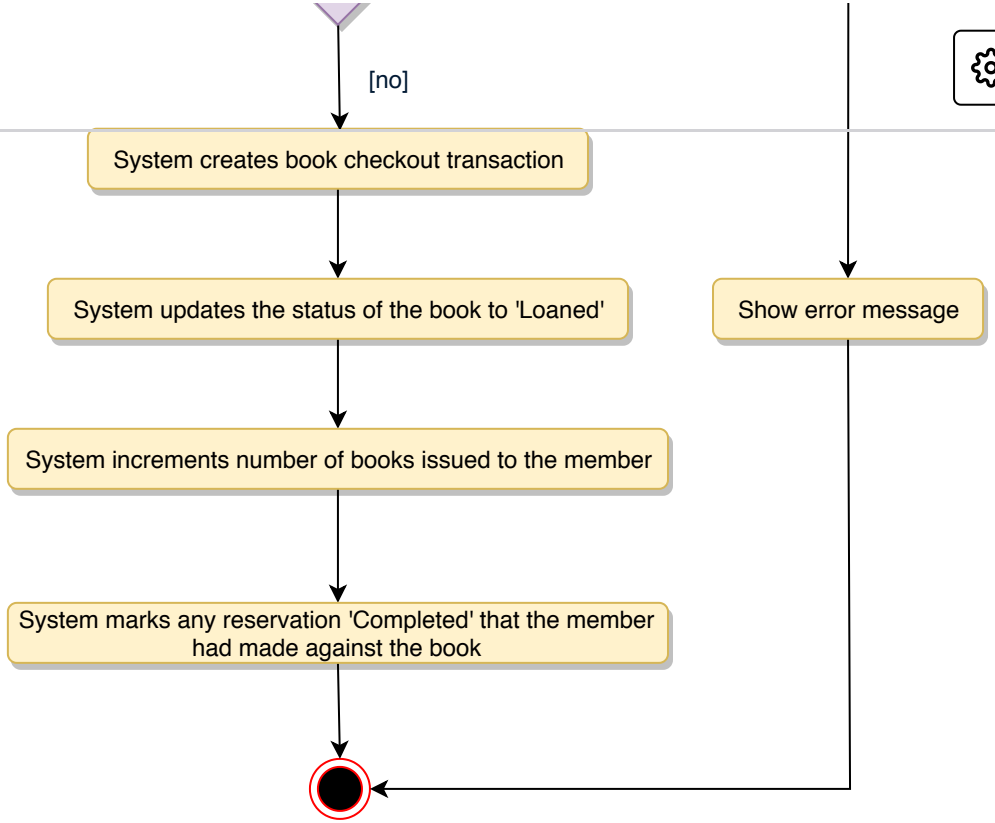
Search Course

oriented-design-interview/7npMYmzm8DA)

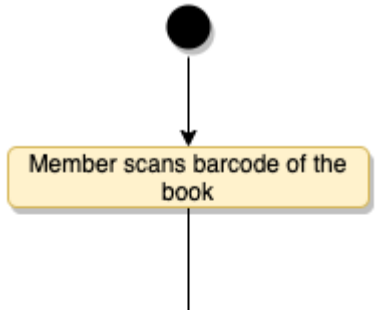
Class Diagram (/courses/grokking-the-object-oriented-design-interview/g7Lw3O0A2Aj)

Sequence diagram (/courses/grokking-the-object-oriented-design-interview/7nX38BMK9NO)

Activity Diagrams (/courses/grokking-the-object-oriented-design-interview/B8RPL3VEl8N)



Return a book: Any library member or librarian can perform this activity. The system will collect fines from members if they return books after the due date. Here are the steps for returning a book:



Explore (/explore)

Tracks (/tracks)

My Courses (/mycourses)

Edpresso (/edpresso)

Refer a Friend (/refer-a-friend)

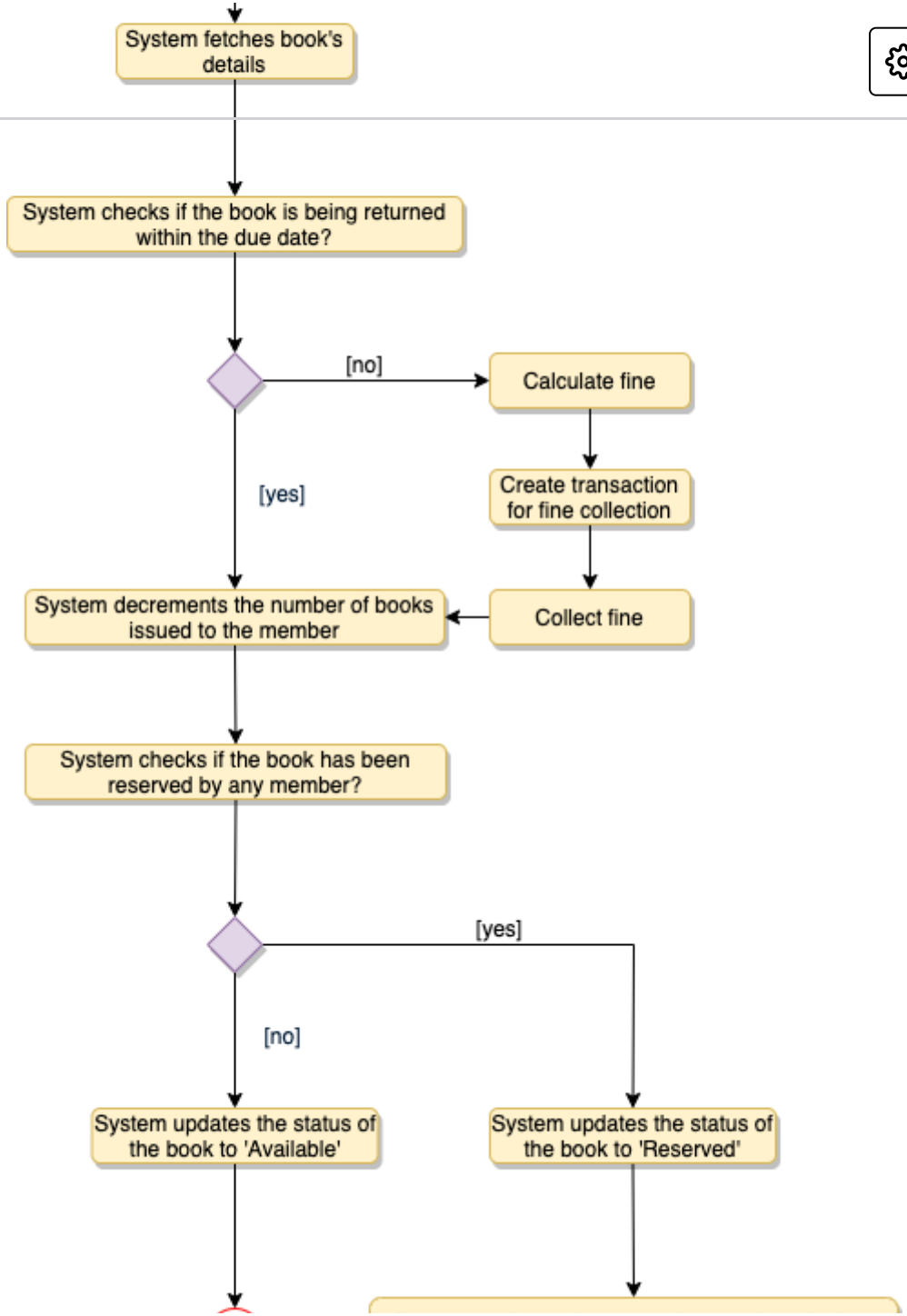
Create

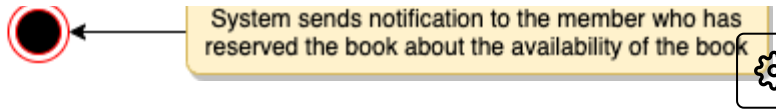
Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

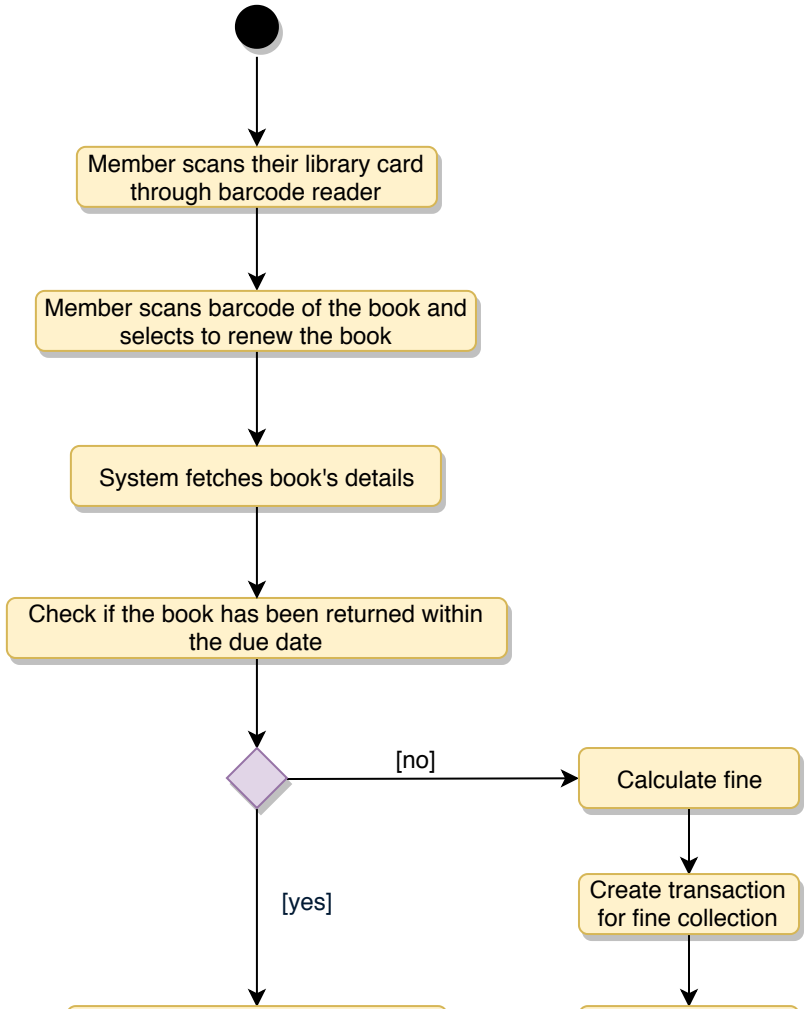
64% completed

Search Course





Renew a book: While renewing (re-issuing) a book, the system will check for fines and see if any other member has not reserved the same book, in that case the book item cannot be renewed. Here are the different steps for renewing a book:



Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

64% completed

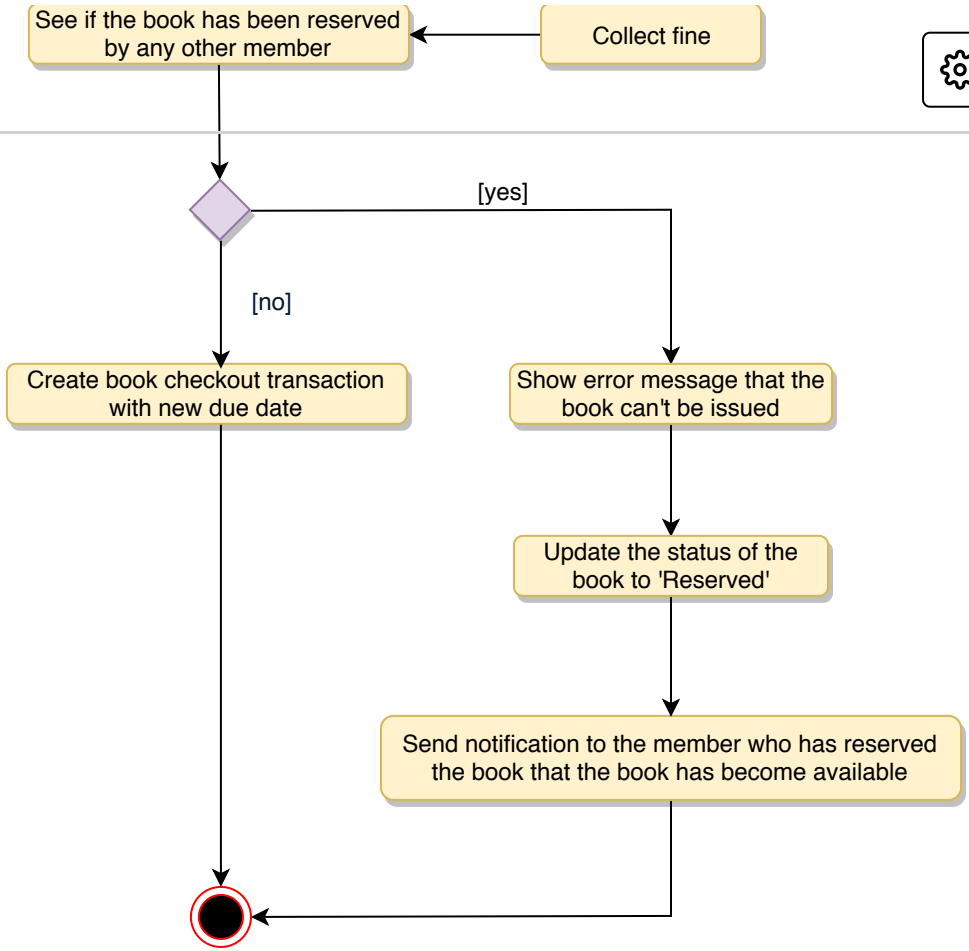
Search Course

oriented-design-interview/7npMYmzm8DA)

Class Diagram (/courses/grokking-the-object-oriented-design-interview/g7Lw3O0A2Aj)

Sequence diagram (/courses/grokking-the-object-oriented-design-interview/7nX38BMK9NO)

Activity Diagrams (/courses/grokking-the-object-oriented-design-interview/B8RPL3VEl8N)



Code

Here is the code for the use cases mentioned above: 1) Check-out a book, 2) Return a book, and 3) Renew a book.

Note: This code only focuses on the design part of the use cases. Since you are not required to write a fully executable code in an

Grokking the Object Oriented Design Interview

interview, you can assume parts of the code to interact with the database, payment system, etc.

Enums and Constants: Here are the required enums, data types, and constants:

Java | Python

```
25  public enum AccountStatus{
26    ACTIVE,
27    CLOSED,
28    CANCELED,
29    BLACKLISTED,
30    NONE
31  }
32
33  public class Address {
34    private String streetAddress;
35    private String city;
36    private String state;
37    private String zipCode;
38    private String country;
39  }
40
41  public class Person {
42    private String name;
43    private Address address;
44    private String email;
45    private String phone;
46  }
47
```

Grokking the Object Oriented Design Interview


```
48  public class Constants {
49    public static final int MAX_BOOKS_ISSUED_TO_A_USER =
50    public static final int MAX_LENDING_DAYS = 10;
51  }
52
```

Account, Member, and Librarian: These classes represent various people that interact with our system:

Java | Python

```
71        // book item has a pending reservation
72        bookItem.updateBookItemStatus(BookStatus.RESERVED);
73        bookReservation.sendBookAvailableNotification();
74      }
75      bookItem.updateBookItemStatus(BookStatus.AVAILABLE);
76    }
77
78    public bool renewBookItem(BookItem bookItem) {
79      this.checkForFine(bookItem.getBarcode());
80      BookReservation bookReservation = BookReservation.fetchReserva
81      // check if this book item has a pending reservation from anot
82      if (bookReservation != null && bookReservation.getMemberId() !
83        ShowError("This book is reserved by another member");
84        member.decrementTotalBooksCheckedout();
85        bookItem.updateBookItemState(BookStatus.RESERVED);
86        bookReservation.sendBookAvailableNotification();
87        return false;
88      } else if (bookReservation != null) {
89        // book item has a pending reservation from this member
90        bookReservation.updateStatus(ReservationStatus.COMPLETED);
91      }
92      BookLending.lendBook(bookItem.getBarCode(), this.getMemberId()
```

Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

64% completed

Search Course

oriented-design-interview/7npMYmzm8DA)

Class Diagram (/courses/grokking-the-object-oriented-design-interview/g7Lw3O0A2Aj)

Sequence diagram (/courses/grokking-the-object-oriented-design-interview/7nX38BMK9NO)

Activity Diagrams (/courses/grokking-the-object-oriented-design-interview/B8RPL3VEl8N)

```
93      bookItem.updateDueDate(LocalDate.now().plusDays(Constants.MAX_
94      return true;
95    }
96  }
97
```

BookReservation, BookLending, and Fine: These classes represent a book reservation, lending, and fine collection, respectively.

Java | Python

```
1  public class BookReservation {
2    private Date creationDate;
3    private ReservationStatus status;
4    private String bookItemBarcode;
5    private String memberId;
6
7    public static BookReservation fetchReservationDetails(String bar
8  }
9
10 public class BookLending {
11   private Date creationDate;
12   private Date dueDate;
13   private Date returnDate;
14   private String bookItemBarcode;
15   private String memberId;
16
17   public static void lendBook(String barcode, String memberId);
18   public static BookLending fetchLendingDetails(String barcode);
19 }
20
21 public class Fine {
```

Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

64% completed

Search Course

oriented-design-interview/7npMYmzm8DA)

Class Diagram (/courses/grokking-the-object-oriented-design-interview/g7Lw3O0A2Aj)

Sequence diagram (/courses/grokking-the-object-oriented-design-interview/7nX38BMK9NO)

Activity Diagrams (/courses/grokking-the-object-oriented-design-interview/B8RPL3VEl8N)

```
21  public class Fine {
22    private Date creationDate;
23    private double bookItemBarcode;
24    private String memberId;
25
26    public static void collectFine(String memberId, long days) {}
27  }
28
```

BookItem: Encapsulating a book item, this class will be responsible for processing the reservation, return, and renewal of a book item.

Java | Python

```
13    private boolean isReferenceOnly;
14    private Date borrowed;
15    private Date dueDate;
16    private double price;
17    private BookFormat format;
18    private BookStatus status;
19    private Date dateOfPurchase;
20    private Date publicationDate;
21    private Rack placedAt;
22
23    public boolean checkout(String memberId) {
24      if(bookItem.getIsReferenceOnly()) {
25        ShowError("This book is Reference only and can't be issued")
26        return false;
27      }
28      if(!BookLending.lendBook(this.getBarCode(), memberId)){
29        return false;
30      }
31      this.updateBookItemStatus(BookStatus.LOANED);
32      return true;
```

Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

64% completed

Search Course

oriented-design-interview/7npMYmzm8DA)

Class Diagram (/courses/grokking-the-object-oriented-design-interview/g7Lw3O0A2Aj)

Sequence diagram (/courses/grokking-the-object-oriented-design-interview/7nX38BMK9NO)

Activity Diagrams (/courses/grokking-the-object-oriented-design-interview/B8RPL3VEl8N)

```
33    }
34  }
35
36  public class Rack {
37    private int number;
38    private String locationIdentifier;
39  }
40
```

Search interface and Catalog: The Catalog class will implement the Search interface to facilitate searching of books.

Java | Python

```
1   public interface Search {
2     public List<Book> searchByTitle(String title);
3     public List<Book> searchByAuthor(String author);
4     public List<Book> searchBySubject(String subject);
5     public List<Book> searchByPubDate(Date publishDate);
6   }
7
8   public class Catalog implements Search {
9     private HashMap<String, List<Book>> bookTitles;
10    private HashMap<String, List<Book>> bookAuthors;
11    private HashMap<String, List<Book>> bookSubjects;
12    private HashMap<String, List<Book>> bookPublicationDates;
13
14    public List<Book> searchByTitle(String query) {
15      // return all books containing the string query in their title
16      return bookTitles.get(query);
17    }
18
19    public List<Book> searchByAuthor(String query) {
```

```
20      // return all books containing the string query in their autho
21      return bookAuthors.get(query);
22    }
23  }
```

Grokking the Object Oriented Design Interview

(/collection/5668639101419520/56922017

64% completed

Search Course

oriented-design-interview/7npMYmzm8DA)

Class Diagram (/courses/grokking-the-object-oriented-design-interview/g7Lw3O0A2Aj)

Sequence diagram (/courses/grokking-the-object-oriented-design-interview/7nX38BMK9NO)

Activity Diagrams (/courses/grokking-the-object-oriented-design-interview/B8RPL3VEl8N)

← Back (/courses/grokking-the-object-oriented-design-interview/B8RPL3VEl8N)

Next → (/courses/grokking-the-object-oriented-design-interview/gxM3gRxmr8Z)

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/c/grokking-the-object-oriented-design-interview-design-gurus/object-oriented-design-case-studies-design-a-library-management-system)