LIBRARY MANAGEMENT SYSTEM

Objective: Designing a 'Library Management System' involving multiple classes: Book, Member, and Library. Implemented multiple constructors, default arguments, public and private fields and complex constraints.

Class Definitions

1. **Class: Book** - Represents a book in the library.

   Attributes:

   1. title (public, string): Title of the book.
   2. author (public, string): Author of the book.
   3. isbn (private, string): Unique identifier for the book.
   4. copiesAvailable (private, int): Number of available copies.
   5. totalCopies (private, int): Total copies of the book in the library.

   Methods:

   1. Constructors:
      - Created a parameterized constructor to initialize all the attributes.
      - A constructor with default arguments for title, author, isbn, copiesAvailable, totalCopies ("UnknownTitle", "UnknownAuthor", "ISBN", 0, 5).
      - One copy constructor to initialize a book using another book instance. Also pass another argument (a new isbn) to this constructor and assign the isbn of the copied book to this new isbn.
   2. Getters and Setters:
      - Getters for isbn, copiesAvailable, and totalCopies.
      - A setter updateCopies(count): increases or decreases the available and the total count of the book ensuring that neither available count or total count becomes less than zero. If the condition is violated then ignore the request and print the statement "Invalid request! Count becomes negative".
   3. Utility Methods:
      - **bool borrowBook()**: Decreases copiesAvailable if a copy is available; returns true if successful and false otherwise (print the message "Invalid request!

Copy of book not available" for this case).

- **bool returnBook()**: Increases copiesAvailable up to totalCopies (if the method is called such that the copiesAvailable exceeds totalCopies then the request should be ignored, an error message should be printed stating "Invalid request! Copy of book exceeds total copies" and the copiesAvailable should not be changed). If return book was successful then return true. Else return false.
- **void printDetails()**: Prints all details of the book. The format should be "title author" (the entities are separated by a single space).

Book Constraints:

1. No two books can have the same isbn. If the constraint is violated then the book which is being entered into the system later is removed from the system (i.e. only the first book which was present in the system persists and all the later books are not added to the system).
2. A book's copiesAvailable should never exceed totalCopies or fall below zero.

2. **Class: Member** – Represents a member of the library.

Attributes:

1. memberID (private, string): Unique identifier for the member.
2. name (public, string): Name of the member.
3. borrowedBooks (private, map<string, int>): Map of isbn to the number of borrowed copies.
4. borrowLimit (private, int): Maximum number of books a member can borrow at a time (by default: 3).

Methods:

1. Constructors:
    - A parameterized constructor to initialize memberID, name, and borrowLimit.
    - A constructor with a default argument for borrowLimit (default value: 3). The other two fields are passed as parameters to the constructor.
2. Utility Methods:
    - **bool borrowBook(isbn)**: Should allow the member to borrow a book if they haven't exceeded their borrow limit. If the constraint is violated then the request is ignored and a message is printed stating "Invalid request! Borrow limit exceeded". Returns true is the book was successfully borrowed. Else returns false.
    - **bool returnBook(isbn)**: Must allow the member to return a book if they've borrowed it. If they haven't borrowed it then ignore the request and print a message stating "Invalid request! Book not borrowed". If return book was successful then return true. Else return false.
    - **void printDetails()**: Prints member details, including a list of borrowed books and

their quantities. The format should be “memberID name isbnOfBook numberOfBorrowedCopies” (the entities are separated by a single space). For each book that is borrowed by the member the same format should be followed on a new line. If the member has not borrowed any book then print nothing.

Member Constraints:

1. A member cannot borrow more than their borrowLimit.
2. A member can borrow the same book multiple times if additional copies are available.

3. **Class: Library** – Represents a library

Attributes:

1. books (private, vector): A list of books in the library.
2. members (private, vector): A list of registered members.

Methods:

1. Utility Methods:
    - **bool addBook(Book&)**: Adds a new book to the library (ensures unique isbn). If the unique isbn constraint is violated then the request is ignored and a message is printed stating “Invalid request! Book with same isbn already exists”. Return true or false respectively if the book was added or not.
    - **bool registerMember(Member&)**: Registers a new member (ensures unique memberID). If the unique memberID constraint is violated then the request is ignored and a message is printed stating “Invalid request! Member with same id already exists”. Return true or false respectively if the book was added or not.
    - **bool borrowBook(string memberID, string isbn)**: Allows a member to borrow a book. First check if the book is available or not. Then check if the borrow limit of the member is within limit or not. Check and print the corresponding error statements in this order only. The error statements are the same as mentioned in the method descriptions of the Book and Member classes above (see public test case 2 for an example).
    - **bool returnBook(string memberID, string isbn)**: Allows a member to return a book. First check if the copies after returning will exceed the total copies or not. Then check if the member ever borrowed that book or not. Check and print the corresponding error statements in this order only. The error statements are the same as mentioned in the method descriptions of the Book and Member classes above (see public test case 3 for an example).
    - **void printLibraryDetails()**: Prints details of all books and members. First the details of the books should be printed with each one on a newline in the format “title author copiesAvailable” (the entities are separated by a single space). Then the details of the members are printed with each one on a newline in the format “memberID name” (the entities are separated by a single space).

Library Constraints:

1. A library cannot have duplicate members or books (based on isbn and memberID).

Constraints

The number of books can be maximum 50 and the number of students can be maximum 150. Each book can have a maximum of 15 copies each. Each member can borrow a maximum of 15 total copies of all books simultaneously.

Some edge cases

Implemented logic to handle edge cases, such as:

- What happens if a member tries to borrow a book with copiesAvailable = 0?
- What if a member tries to return a book they never borrowed?
- You need to handle a scenario where the library runs out of a specific book.

Note

If a book is borrowed multiple times by the same member then the book has to be returned the same number of times by the member (e.g. if there are two borrow commands for the same book by the same member and both of them are successful then for returning the book there will be two separate return commands).

Input Format

The commands for each entity and operation will have a similar pattern like:

Entity/operation
Details of the entity/operation

Book

A new book will be added by the "**Book**" word followed by the details. There can be three scenarios for books as described below.

Book
Case 1 format: Title Author ISBN CopiesAvailable TotalCopies
Case 2 format: None
Case 3 format: ExistingBook OldIsbn NewIsbn

Increasing the count of a book

The count of an existing book will be updated by the "**UpdateCopiesCount**" word followed by the details. The format will be as follows:

UpdateCopiesCount
Isbn NewCount

Member

A new member will be added by the "**Member"** word followed by the details. There can be two scenarios for members as described below.

Member
Case 1 format: MemberId Name BorrowLimit
Case 2 format: NoBorrowLimit MemberId Name

Borrow Operation

Borrow book operation will be given by the "**Borrow**" word followed by the details. The format is as follows:

Borrow
MemberId isbn

Return Operation

Return book operation will be given by the "**Return**" word followed by the details. The format is as follows:

Return
MemberId isbn

Print book details

The details of the book to be printed is as follows:

PrintBook
isbn

Print member details

The details of the member to be printed is as follows:

PrintMember
memberId

Print library details

The details of the library to be printed is as follows:

PrintLibrary

End of input

The input will end with the "**Done**" keyword.

Done

Output Format

The outputs for the methods are mentioned within the description of the respective function within quotes. The format is also mentioned there. See the sample test cases for more clarification.

Output of print book

The format for this will be:

title author

Output of print member

For each of the book borrowed by the member, the format for this will be:

memberID name isbnOfBook numberOfBorrowedCopies

If no books are borrowed by the member, then nothing will be printed for this operation.

Output of print library

The format for the books will be:

title author copiesAvailable

The format for the members will be:

memberID name