System Requirements Specification Index

For

Library Management Console InMemory- Beginner

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Table of Contents

Library Management Console
System Requirements Specification

1 Project Abstract

Library Management Console Application is a pure java application with Java collection, where it allows to manage the books and issue the books from the library. The Library Management System empowers users to perform CRUD (Create, Read, Update, Delete) operations on books. Users can create new book entries, update existing book and subject information, delete books and many more relevant operations.

2 Business Requirements:

Screen Name	Console input screen
Problem Statement	1. User needs to enter into the application. 2. The user should be able to do the particular operations 3. The console should display the menu 1) Add Book 2) Get Book by Name 3) Issue a Book 4) Check Availability 5) List Borrowed Books 6) Update Book 7) Get All Books 8) Exit

2.1 LibraryManagementApp Class - Method Descriptions

Method	Task	Implementation Details	Return Value

`main(String[] args)`	Entry point for the application	- **Initialize Scanner and Inventory:** Create instances of `Scanner` and `Inventory`.
		- **Display Menu:** Show the console menu with options for adding, updating, viewing, and issuing books, etc.
		- **Handle User Input:** Use `switch` to call corresponding methods based on user choice.
		- **Exit Option:** If the user selects "Exit" option, close `Scanner` and terminate the program.
			Returns: `void` (Runs indefinitely until user selects exit).
			Example Output:
			`Options:`
			`1. Add Book`
			`2. Get Book by Name`
			`3. Issue a Book`
			`4. Check Availability`
			`5. List Borrowed Books`
			`6. Update Book`
			`7. Get All Books`
			`8. Exit`
			`Enter your choice:`

`addBook(Inventory inventory)`	Add a new book to the inventory	- **Prompt for Details:** Ask for ISBN, Title, Author, and Publisher.
		- **Create Book Object:** Create a `Book` object with the provided details and set `available = true`.
		- **Call Inventory Method:** Call `inventory.addBook(newBook)`. Handle `ISBNAlreadyExistsException` if thrown.
			Returns: `void` (Prints success message).
			Throws: `ISBNAlreadyExistsException` if ISBN already exists with message: "Error: " + e.getMessage().
			Example Output:
			`"Book added successfully."`
			or
			`"Error: Book with the same ISBN already exists."`

`getBookByName(Inventory inventory)`	Search for a book by name	- **Prompt for Book Name:** Ask the user to enter the book's name.
		
		- **Call Inventory Method:** Call `inventory.getBookByName(name)`, which returns an `Optional<Book>`.
		
		- **Handle Optional:** If present, display book details including Title, Author, Publisher, ISBN, and Status.
		
		- **Status Check:** If `book.isAvailable()` is `true`, print `"Status: Available"`, otherwise print `"Status: Issued"` along with `Issued Date` and `Due Date`.
		
		- **Book Not Found:** If `Optional` is empty, print `"Book not found."`
			Returns: `void` (Displays book details or not found message).
			
			Example Output:
			
			If the book is **available**:
			
			`Book found: Title: Java Programming Author: John Doe Publisher: Pearson`
			
			`ISBN: 123456789 Status: Available`
			
			If the book is **issued**:
			
			`Book found: Title: Java Programming Author: John Doe Publisher: Pearson`
			
			`ISBN: 123456789 Status: Issued Issued Date: 2024-03-01`
			
			`Due Date: 2024-03-15`
			
			If not found:
			
			`"Book not found."`

`issueBook(Inventory inventory)`	Issue a book to a user	- **Prompt for Details:** Ask for ISBN, Username, and Due Date. - **Check Availability:** If the book is not available, print: "The book with ISBN " + isbn + " is not available for issuance." and return.	**Returns:** `void` (Prints success or error message). **Example Output:** `"Book issued successfully to`

		- **Issue Book:** If available, call `inventory.issueBook()` and display a success or failure message as : "Failed to issue the book. Please check the book availability and user details.".	`user [username]."`
			If not available:
			`"The book with ISBN [isbn] is not available for issuance."`
			If error:
			`"Failed to issue the book. Please check the book availability and user details."`

`checkAvailability(Inventory inventory)`	Check if a book is available	- **Prompt for ISBN:** Ask the user to enter the ISBN of the book.
		- **Call Inventory Method:** Call `inventory.isBookAvailable(isbn)` to check availability.
		- **Display Result:** Print whether the book is available or not.
			Returns: `void` (Displays availability status).
			Example Output:
			`"The book with ISBN [isbn] is available."`
			If not available:
			`"The book with ISBN [isbn] is not available."`

`listBorrowedBooks(Inventory inventory)`	List all borrowed books	- **Call Inventory Method:** Call `inventory.getBorrowedBooks()` to retrieve a list of borrowed books.
		- **Check List:** If the list is empty, print `"No books are currently borrowed."`
		- **Display Books:** Loop through the list and print each borrowed book's details.
			Returns: `void` (Prints borrowed books or empty message).
			Example Output:
			`Borrowed Books:`
			`Title: Java Programming`
			`ISBN: 123456789`
			`Username: johndoe`
			`Due Date: 2024-03-15`

			If no books: `"No books are currently borrowed."`

`updateBook(Inventory inventory)`	Update details of an existing book	- **Prompt for ISBN:** Ask for the ISBN of the book to update.
		- **Find Book:** Find the book by ISBN.
		- **Check Existence:** If no book is found, print `"No book found with ISBN [isbn]."`
		- **Prompt for Updates:** Ask for updated Title, Author, and Publisher.
		- **Update Book:** Call `inventory.updateBook(updatedBook)`.
		- **Display Success Message:** Print `"Book updated successfully."`
			Returns: `void` (Updates the book).
			Example Output:
			`"Book updated successfully."`
			If not found:
			`"No book found with ISBN [isbn]."`

`getAllBooks(Inventory inventory)`	Display all books in the inventory	- **Call Inventory Method:** Call `inventory.getAllBooks()` to retrieve a list of all books.
		- **Check List:** If the list is empty, print `"No books available."`
		- **Display Books:** Loop through the list and print details of each book.
			Returns: `void` (Prints all books or empty message).
			Example Output:
			`All Books:`
			`Title: Java Programming`
			`ISBN: 123456789`
			`Author: John Doe`
			`Publisher: Pearson`
			`Available: Yes`
			If no books:
			`"No books available."`

2.2 Inventory Class - Method Descriptions

Method	Task	Implementation Details	Return Value
`addBook(Book book)`	Add a new book to the inventory	- **Check for Existing ISBN:** Check if a book with the same ISBN already exists or not. - **Throw Exception:** If ISBN already exists, throw `ISBNAlreadyExistsException` with the message: `"Book with the same ISBN already exists."`. - **Add to List:** If the ISBN is unique, add the `Book` object to the `books` list.	**Returns:** `void` (Adds the book to inventory). **Throws:** `ISBNAlreadyExistsException` if the ISBN already exists.
`getBookByName(String name)`	Retrieve a book by its name	- **Search by Name:** Find a `Book` where the title contains the given `name` in parameter(case-insensitive). - **Return as Optional:** Return the result wrapped in an `Optional<Book>`.	**Returns:** `Optional<Book>`
`updateBook(Book updatedBook)`	Update details of an existing book	- **Find Book by ISBN:** Find a book with the same ISBN fetched from a passed `Book` object in parameter. - **Update Details:** If found, update `title`, `author`, `publisher`, `available`, `issuedDate`, `dueDate`, and `username` fields. - **Return Updated Book:** Return the updated `Book` object. - **Handle Absence:** If no book is found with the ISBN, return `null`.	**Returns:** `Book` (Updated book details). **Returns:** `null` if no book is found with the given ISBN.

`getAllBooks()`	Retrieve all books in the inventory	- **Return List:** Return the `books` list containing all `Book` objects.	**Returns:** `List<Book>` (All books in the inventory).
`issueBook(String isbn, String username, LocalDate dueDate)`	Issue a book to a user	- **Check Availability:** Find a `Book` with the matching ISBN (passed as parameter that is currently available). - **Update Book Status:** If found, set `available` to `false`, update `issuedDate` to `LocalDate.now()`, set `dueDate` to the provided date, and record the `username`. - **Return Status:** Return `true` if the book is successfully issued, otherwise return `false`.	**Returns:** `boolean` (`true` if issued successfully, `false` otherwise).
`isBookAvailable(String isbn)`	Check availability of a book	- **Check Availability:** Check if any `Book` has the given ISBN and is marked as `available`. - **Return Result:** Return `true` if the book is available, otherwise return `false`.	**Returns:** `boolean` (`true` if available, `false` if not).
`getBorrowedBooks()`	List all borrowed books	- **Filter Borrowed Books:** Get all borrowed `Book` objects as a list. - **Return List:** Return a list of all borrowed books.	**Returns:** `List<Book>` (All borrowed books).

3 Constraints

3.1 Book Constraints

- When adding a book with an ISBN that already exists in the inventory, the method should throw an `ISBNAlreadyExistsException` with the message:

  "Book with the same ISBN already exists."

- When trying to fetch a book by name that does not exist in the inventory, the method should return an empty `Optional<Book>` .
- When trying to update a book that does not exist in the inventory, the method should return `null`.

3.2 Common Constraints

- Take console input of number of books: (n)
- Take input of details of each book and store it in a collection.
- Take input of details of books to be issued (only 1 book at a time).
- Take input of details of book to issue and store in a collection.
- Show the books stock remained after issuing books.

4 Template Code Structure

4.1 Package: com.elibrary

Resources

Class/Interface	Description	Status
LibraryManagementApp.java(class)	**This represents bootstrap class i.e class with Main method, that shall contain all console interaction with the user.**	**Partially implemented**

4.2 Package: com.iiht.training.elibrary.model

Resources

Class/Interface	Description	Status
Book (class)	• This class contains all the properties of the Book class.	Already implemented.

4.3 Package: com.iiht.training.elibrary.inventory

Resources

Class/Interface	Description	Status
Inventory (class)	• This class contains all the methods which are used to write the business logic for the application • You can create any number of private methods in the class	Partially implemented.

4.4 Package: com.iiht.training.elibrary.exception

Resources

Class/Interface	Description	Status
ISBNAlreadyExistsException (Class)	• Custom Exception to be thrown when trying to add a book for which ISBN is already exists	Already created.
AlreadyIssuedException (Class)	• Custom Exception to be thrown when trying to issue a book which is already issued.	Already created.

5 Execution Steps to Follow

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal.**
3. **This editor Auto Saves the code.**
4. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
5. **To run your project use command:**
   mvn clean install exec:java -Dexec.mainClass="com.elibrary.LibraryManagementApp"
7. **To test your project, use the command**
   mvn test