**Digital Library Book Management System:**

**----------------------------------------------------------------------------**

**Step 1: Understanding the Requirements**

The system needs to:

- Add books with ID, Title, Author, Genre, and Availability.

- View all books.

- Search for books by ID or Title.

- Update book details.

- Delete a book record.

- Exit the system.

Constraints:

- Book ID must be unique.

- Title and Author cannot be empty.

- Availability should be "Available" or "Checked Out".

**-------------------------------------------------------------------------------**

**Step 2: Designing the Solution**

We need:

1. A Book class to represent book objects.

2. A DigitalLibrary class that contains:

- A HashMap<String, Book> to store books efficiently.

- A Scanner for user input.

- Methods for adding, viewing, searching, updating, and deleting books.

**---------------------------------------------------------------------------------**

**Step 3: Implementing the Book Class**

//Book.java

class Book {

private String id;

private String title;

private String author;

private String genre;

private String availability;

//This class encapsulates a book’s properties.

//Constructor:

public Book(String id, String title, String author, String genre, String availability) {

//- Ensures ID, Title, and Author are non-empty.

//- Ensures availability is either "Available" or "Checked Out".

//Getters and Setters:

public String getId() { return id; }

public String getTitle() { return title; }

public String getAuthor() { return author; }

public String getGenre() { return genre; }

public String getAvailability() { return availability; }

//- Used to retrieve book details.

public void setAvailability(String availability) {

//- Ensures only valid availability values.

**----------------------------------------------------------------------------------**

**////Step #4: Implementing the DigitalLibrary Class**

//This class acts as the main driver of the program.

//#. main() Method:

public static void main(String[] args) {

//- Displays a menu with options (1-6).

//- Calls respective methods based on user input.

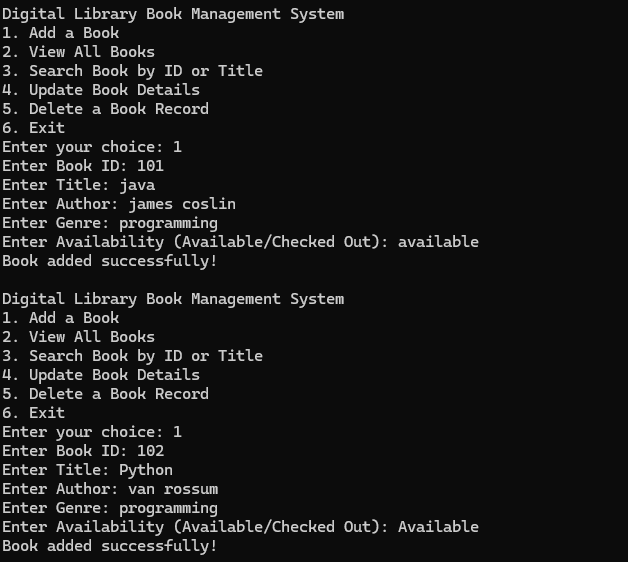
//1. addBook() Method

private static void addBook() {

//- Takes user input for book details.

//- Checks if ID is unique before adding.

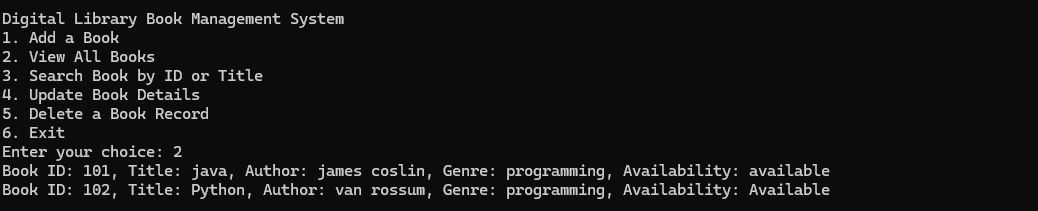
//-Stores the book in the HashMap.



//2. viewAllBooks() Method

private static void viewAllBooks() {

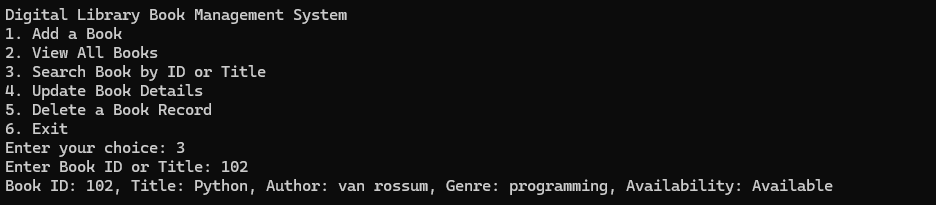
//-Loops through HashMap to display all books.



//3. searchBook() Method

private static void searchBook() {

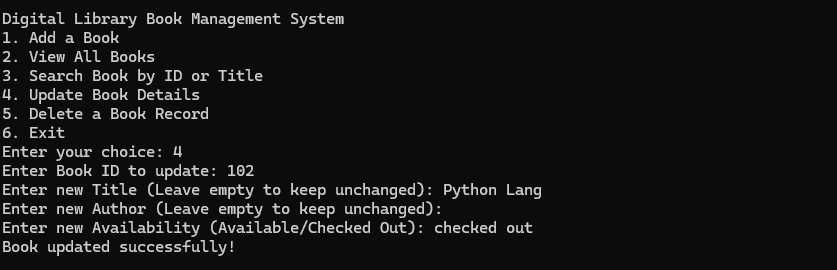
//Finds book by ID or Title.



//4.updateBook() Method

private static void updateBook() {

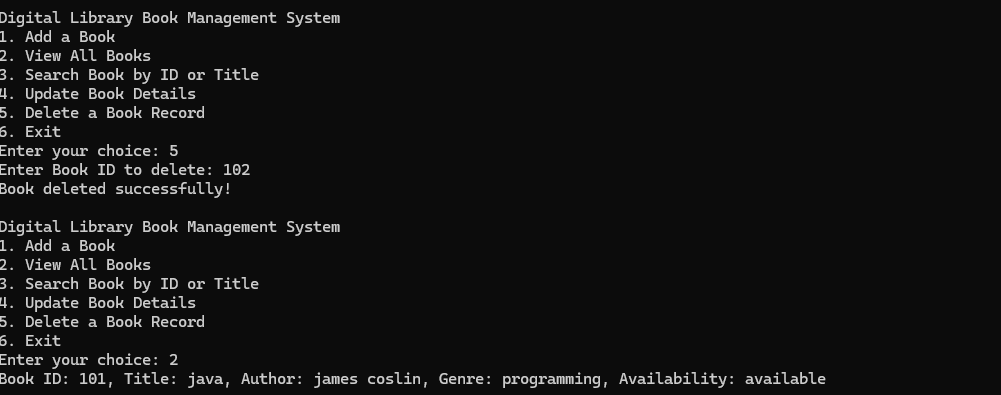
//- Allows updating Title, Author, or Availability.



//5. deleteBook() Method

private static void deleteBook() {

//- Removes book from HashMap using ID.



**------------------------------------------------------------------------------------**

**Step 5: Running the Application**

//Compile: javac DigitalLibrary.java

//Run: java DigitalLibrary`

//Example Interaction:

Digital Library Book Management System

1. Add a Book

2. View All Books

3. Search Book by ID or Title

4. Update Book Details

5. Delete a Book Record

6. Exit

Enter your choice: 1

Enter Book ID: 101

Enter Title: Java Programming

Enter Author: James Gosling

Enter Genre: Programming

Enter Availability (Available/Checked Out): Available

Book added successfully!

**-------------------------------------------------------------------------------------**

**Step 6: Handling Edge Cases**

- Empty input validation for Title and Author.

- Invalid availability input handling.

- Unique Book ID check before adding.

- Proper messages for search/update/delete if book not found.