**Exercise 2: Implementing Dependency Injection**

Scenario:

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

* **Introduction**

This document outlines the steps and modifications required to implement Dependency Injection (DI) in the Library Management Application. DI is a key feature of the Spring Framework, enabling better modularity, testability, and maintainability of your application.

* **Steps to Implement Dependency Injection**

**Step 1: Modify the XML Configuration**

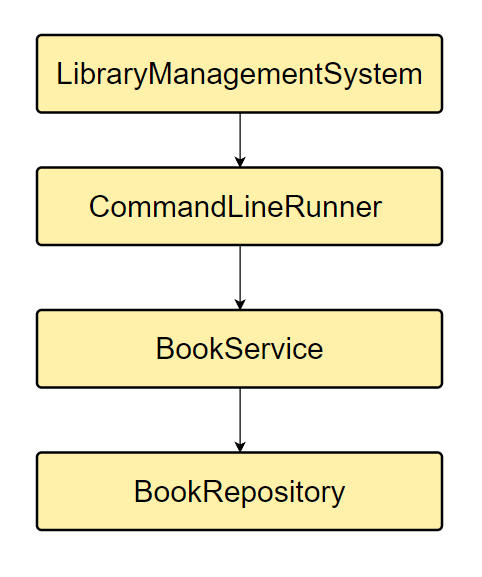
* The bookRepository bean is defined to create an instance of BookRepository.
* The bookService bean is configured to inject the bookRepository bean into the BookService using the <property> tag.

**Step 2: Update the BookService Class**

* The setBookRepository method allows Spring to inject the BookRepository instance into BookService.
* This method is automatically called by Spring during bean initialization, as configured in applicationContext.xml.

**Step 3: Test the Configuration**

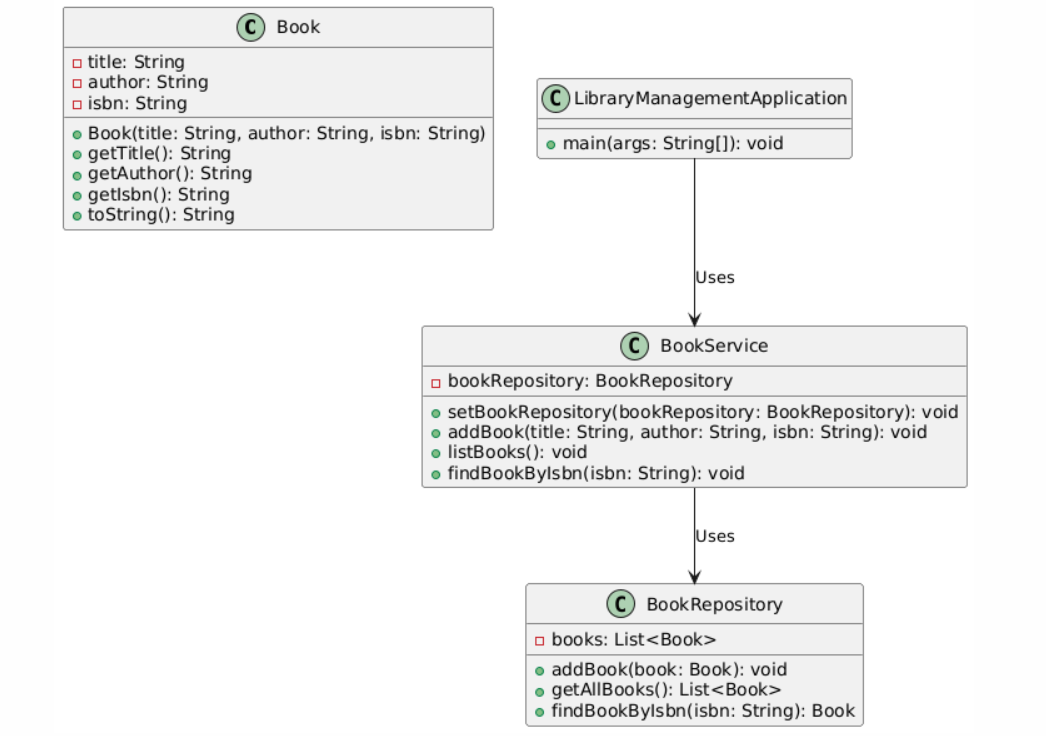
* The application should run without errors.
* The BookService should be correctly using the injected BookRepository to manage the library's books.
* **FLOWCHART of the program:**



**Explanation:**

* **Start:** The entry point of the application.
* **Book Class:** Defines the data model for a book, including attributes such as title, author, and isbn.
* **BookRepository Class:** Manages a list of Book objects and provides methods for adding, retrieving, and finding books by ISBN.
* **BookService Class:** Provides higher-level business logic and interacts with BookRepository. This class uses dependency injection to receive an instance of BookRepository.
* **LibraryManagementApplication:** The main application class that sets up the Spring context, invokes methods on BookService, and manages user interactions.
* **ApplicationContext:** The Spring container that manages the lifecycle and dependencies of the beans (BookRepository and BookService).
* **End:** Represents the conclusion of the application's flow.

**CLASS DIAGRAM:**



**Explanation:**

* The class diagram shows the structure of the Book, BookRepository, BookService, and LibraryManagementApplication classes.
* **Book:** Represents the entity model for books in the library.
* **BookRepository:** Manages a collection of Book objects. It is responsible for adding, retrieving, and finding books by ISBN.
* **BookService:** Provides business logic for managing books. It uses BookRepository to perform operations. The dependency is injected via Spring's IoC container.
* **LibraryManagementApplication:** The entry point of the application that interacts with the BookService to manage the library system.
* The arrows represent dependencies between the classes, where LibraryManagementApplication depends on BookService, and BookService depends on BookRepository.
* **Output of the code:** it prints the books name

