**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.

**Steps:**

1. **Modify the XML Configuration:**
   * Update **applicationContext.xml** to wire **BookRepository** into **BookService**.
2. **Update the BookService Class:**
   * Ensure that **BookService** class has a setter method for **BookRepository**.
3. **Test the Configuration:**
   * Run the **LibraryManagementApplication** main class to verify the dependency injection.

**Solution:**

LibraryManagementDIApplication.java:

package com.library;  
  
import com.library.service.BookService;  
import com.library.repository.BookRepository;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryManagementDIApplication {  
 public static void main(String[] args) {  
 System.*out*.println("=== Starting Library Management DI Application ===");  
 System.*out*.println("Initializing Spring Application Context...\n");  
  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 System.*out*.println("\n=== Spring Context Loaded Successfully ===");  
  
 BookService bookService = (BookService) context.getBean("bookService");  
  
 System.*out*.println("\n=== Testing Dependency Injection ===");  
 if (bookService.getBookRepository() != null) {  
 System.*out*.println("✓ Dependency Injection successful!");  
 System.*out*.println("✓ BookRepository is properly injected into BookService");  
 } else {  
 System.*out*.println("✗ Dependency Injection failed!");  
 }  
  
 System.*out*.println("\n=== Testing Library Management Operations ===");  
  
 bookService.displayAllBooks();  
  
 bookService.getLibraryStats();  
  
 System.*out*.println("\n=== Testing Add Book Functionality ===");  
 bookService.addBook("Spring Boot in Action");  
 bookService.addBook("Microservices Patterns");  
 bookService.addBook("Spring Boot in Action"); // Duplicate test  
  
 bookService.displayAllBooks();  
  
 System.*out*.println("\n=== Testing Search Functionality ===");  
 bookService.searchBook("Spring");  
 bookService.searchBook("Java");  
 bookService.searchBook("Python");  
  
 System.*out*.println("\n=== Testing Remove Book Functionality ===");  
 bookService.removeBook("Clean Code");  
 bookService.removeBook("Non-existent Book");  
  
 bookService.displayAllBooks();  
 bookService.getLibraryStats();  
  
 System.*out*.println("\n=== Testing Bean Scopes ===");  
 BookService bookService2 = (BookService) context.getBean("bookService");  
 System.*out*.println("BookService instances are same: " + (bookService == bookService2));  
  
 BookRepository repository = (BookRepository) context.getBean("bookRepository");  
 System.*out*.println("Repository instances are same: " + (bookService.getBookRepository() == repository));  
  
 ((ClassPathXmlApplicationContext) context).close();  
  
 System.*out*.println("\n=== DI Application Completed Successfully! ===");  
 System.*out*.println("Spring context closed properly.");  
 }  
}

**BookService.java:**

package com.library.service;  
  
import com.library.repository.BookRepository;  
import java.util.List;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public BookService() {  
 System.*out*.println("BookService bean created");  
 }  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 System.*out*.println("BookRepository injected into BookService via setter method");  
 }  
  
 public BookRepository getBookRepository() {  
 return bookRepository;  
 }  
  
 public void addBook(String bookTitle) {  
 if (bookRepository == null) {  
 System.*out*.println("Error: BookRepository not injected");  
 return;  
 }  
  
 if (bookTitle != null && !bookTitle.trim().isEmpty()) {  
 if (bookRepository.bookExists(bookTitle)) {  
 System.*out*.println("Book already exists: " + bookTitle);  
 } else {  
 bookRepository.addBook(bookTitle);  
 System.*out*.println("BookService: Successfully added book - " + bookTitle);  
 }  
 } else {  
 System.*out*.println("BookService: Invalid book title provided");  
 }  
 }  
  
 public void removeBook(String bookTitle) {  
 if (bookRepository == null) {  
 System.*out*.println("Error: BookRepository not injected");  
 return;  
 }  
  
 boolean removed = bookRepository.removeBook(bookTitle);  
 if (removed) {  
 System.*out*.println("BookService: Successfully removed book - " + bookTitle);  
 } else {  
 System.*out*.println("BookService: Failed to remove book - " + bookTitle);  
 }  
 }  
  
 public List<String> getAllBooks() {  
 if (bookRepository == null) {  
 System.*out*.println("Error: BookRepository not injected");  
 return null;  
 }  
 return bookRepository.getAllBooks();  
 }  
  
 public void displayAllBooks() {  
 if (bookRepository == null) {  
 System.*out*.println("Error: BookRepository not injected");  
 return;  
 }  
  
 System.*out*.println("\n=== Library Management System ===");  
 bookRepository.displayBooks();  
 System.*out*.println("Total books in library: " + bookRepository.getTotalBooks());  
 System.*out*.println("================================");  
 }  
  
 public void searchBook(String keyword) {  
 if (bookRepository == null) {  
 System.*out*.println("Error: BookRepository not injected");  
 return;  
 }  
  
 System.*out*.println("\nSearching for books containing: '" + keyword + "'");  
 List<String> matchingBooks = bookRepository.searchBooks(keyword);  
  
 if (matchingBooks.isEmpty()) {  
 System.*out*.println("No books found matching: " + keyword);  
 } else {  
 System.*out*.println("Found " + matchingBooks.size() + " book(s):");  
 for (int i = 0; i < matchingBooks.size(); i++) {  
 System.*out*.println((i + 1) + ". " + matchingBooks.get(i));  
 }  
 }  
 }  
  
 public void getLibraryStats() {  
 if (bookRepository == null) {  
 System.*out*.println("Error: BookRepository not injected");  
 return;  
 }  
  
 System.*out*.println("\n=== Library Statistics ===");  
 System.*out*.println("Total books: " + bookRepository.getTotalBooks());  
 System.*out*.println("Repository status: " + (bookRepository.getTotalBooks() > 0 ? "Active" : "Empty"));  
 System.*out*.println("========================");  
 }  
}

**BookRepository.java:**

package com.library.repository;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class BookRepository {  
 private List<String> books;  
  
 public BookRepository() {  
 this.books = new ArrayList<>();  
 // Initialize with some sample books  
 books.add("Spring in Action");  
 books.add("Java: The Complete Reference");  
 books.add("Clean Code");  
 books.add("Effective Java");  
 books.add("Design Patterns");  
 System.*out*.println("BookRepository bean created and initialized with sample books");  
 }  
  
 public void addBook(String bookTitle) {  
 if (bookTitle != null && !bookTitle.trim().isEmpty()) {  
 books.add(bookTitle);  
 System.*out*.println("Book added to repository: " + bookTitle);  
 } else {  
 System.*out*.println("Cannot add book: Invalid title");  
 }  
 }  
  
 public List<String> getAllBooks() {  
 return new ArrayList<>(books);  
 }  
  
 public boolean removeBook(String bookTitle) {  
 boolean removed = books.remove(bookTitle);  
 if (removed) {  
 System.*out*.println("Book removed from repository: " + bookTitle);  
 } else {  
 System.*out*.println("Book not found in repository: " + bookTitle);  
 }  
 return removed;  
 }  
  
 public int getTotalBooks() {  
 return books.size();  
 }  
  
 public void displayBooks() {  
 System.*out*.println("Books in repository:");  
 if (books.isEmpty()) {  
 System.*out*.println("No books available");  
 } else {  
 for (int i = 0; i < books.size(); i++) {  
 System.*out*.println((i + 1) + ". " + books.get(i));  
 }  
 }  
 }  
  
 public boolean bookExists(String bookTitle) {  
 return books.contains(bookTitle);  
 }  
  
 public List<String> searchBooks(String keyword) {  
 List<String> matchingBooks = new ArrayList<>();  
 for (String book : books) {  
 if (book.toLowerCase().contains(keyword.toLowerCase())) {  
 matchingBooks.add(book);  
 }  
 }  
 return matchingBooks;  
 }  
}

**applicationContext.xml:**

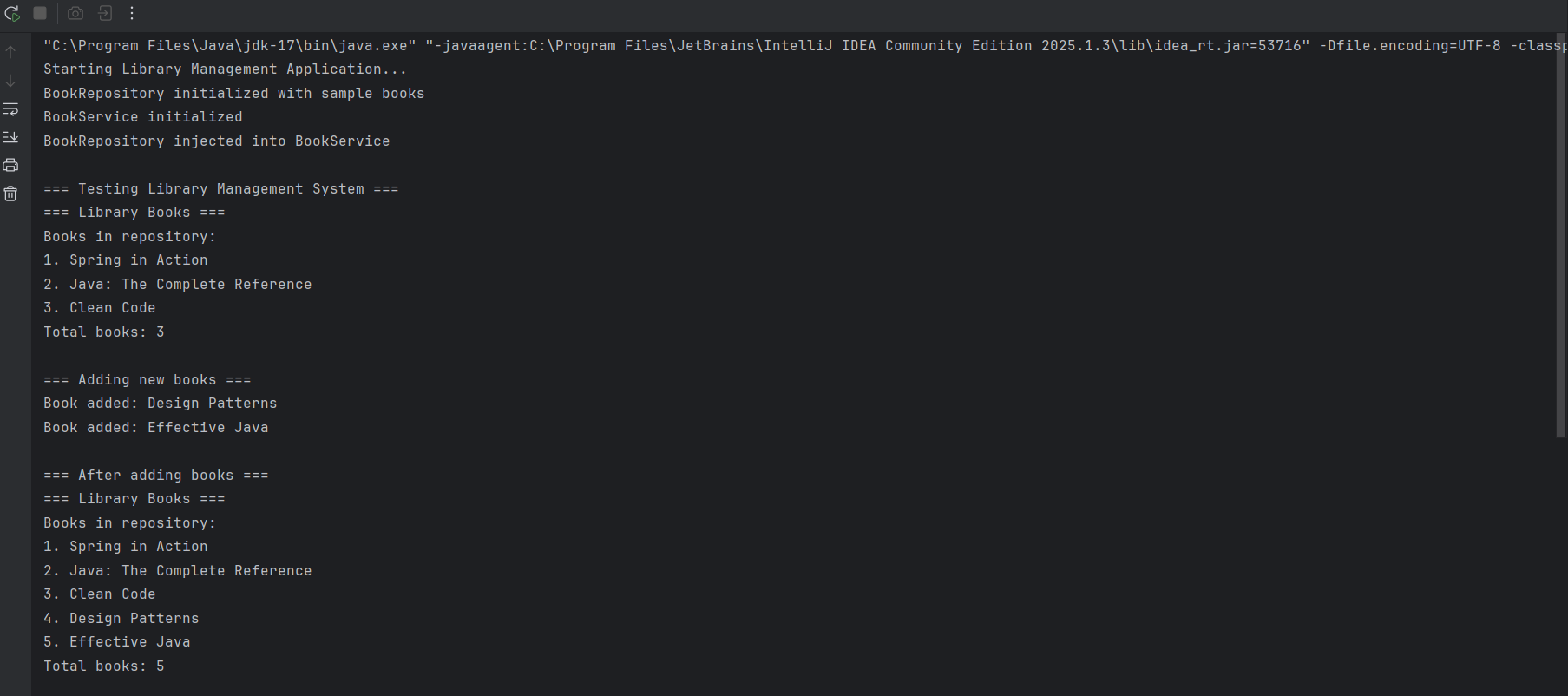
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- BookRepository Bean Definition -->  
 <bean id="bookRepository"  
 class="com.library.repository.BookRepository">  
 <!-- No properties needed as it's initialized in constructor -->  
 </bean>  
  
 <!-- BookService Bean Definition with Dependency Injection -->  
 <bean id="bookService"  
 class="com.library.service.BookService">  
 <!-- Setter-based dependency injection -->  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
 <!-- Alternative: You can also define multiple BookService beans  
 with different configurations if needed -->  
  
 <!-- Example of bean with different scope (uncomment if needed)  
 <bean id="bookServicePrototype"  
 class="com.library.service.BookService"  
 scope="prototype">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
 -->  
  
</beans>

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.library</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>LibraryManagement</name>  
 <description>Spring-based Library Management System with Dependency Injection</description>  
  
 <properties>  
 <maven.compiler.source>11</maven.compiler.source>  
 <maven.compiler.target>11</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <spring.version>5.3.21</spring.version>  
 </properties>  
  
 <dependencies>  
 <!-- Spring Core -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
  
 <!-- Spring Context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
  
 <!-- Spring Beans -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-beans</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
  
 <!-- JUnit for testing -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>11</source>  
 <target>11</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Output:**

A black rectangle with white dots

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

A black rectangle with white dots

AI-generated content may be incorrect.

A black rectangle with white dots

AI-generated content may be incorrect.