**Exercise 7: Implementing Constructor and Setter Injection**

**Scenario:**

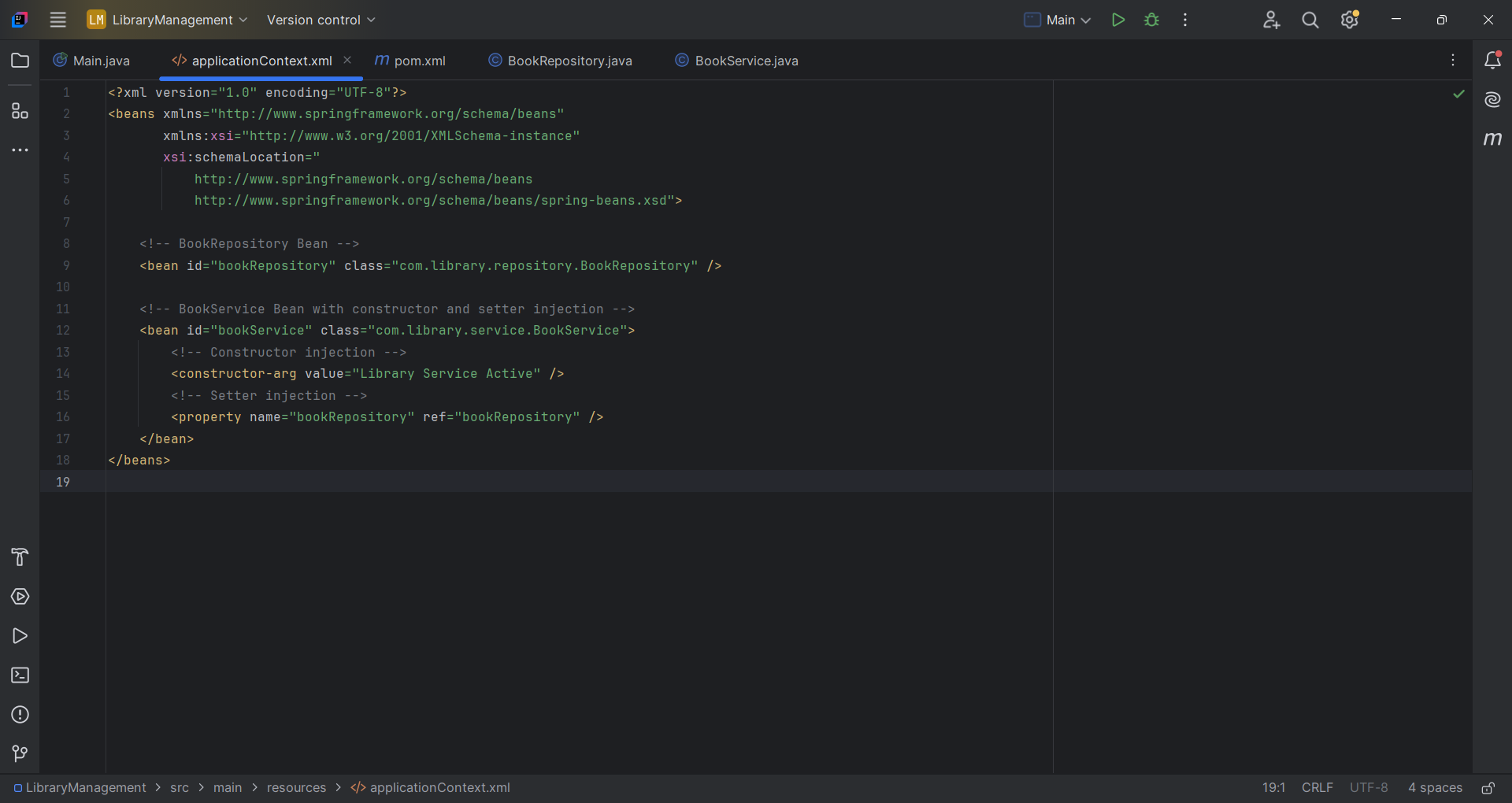
The library management application requires both constructor and setter injection for better control over bean initialization.

**Steps:**

1. **Configure Constructor Injection:**
   * Update applicationContext.**xml** to configure constructor injection for **BookService**.
2. **Configure Setter Injection:**
   * Ensure that the **BookService** class has a setter method for **BookRepository** and configure it in **applicationContext.xml**.
3. **Test the Injection:**
   * Run the **LibraryManagementApplication** main class to verify both constructor and setter injection.

**Solution:**

* + 1. **Configure Constructor Injection:**

****

* + 1. **Configure Setter Injection:**

**//file name BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private String statusMessage; // Constructor-injected

private BookRepository bookRepository; // Setter-injected

// Constructor for injection

public BookService(String statusMessage) {

this.statusMessage = statusMessage;

}

// Setter for BookRepository

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void listBooks() {

System.out.println("BookService Status: " + statusMessage);

bookRepository.displayBooks();

}

}

**//file name BookRepository.java**

package com.library.repository;

public class BookRepository {

public void displayBooks() {

System.out.println("BookRepository: Displaying books...");

}

}

3. **Test the Injection:**

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

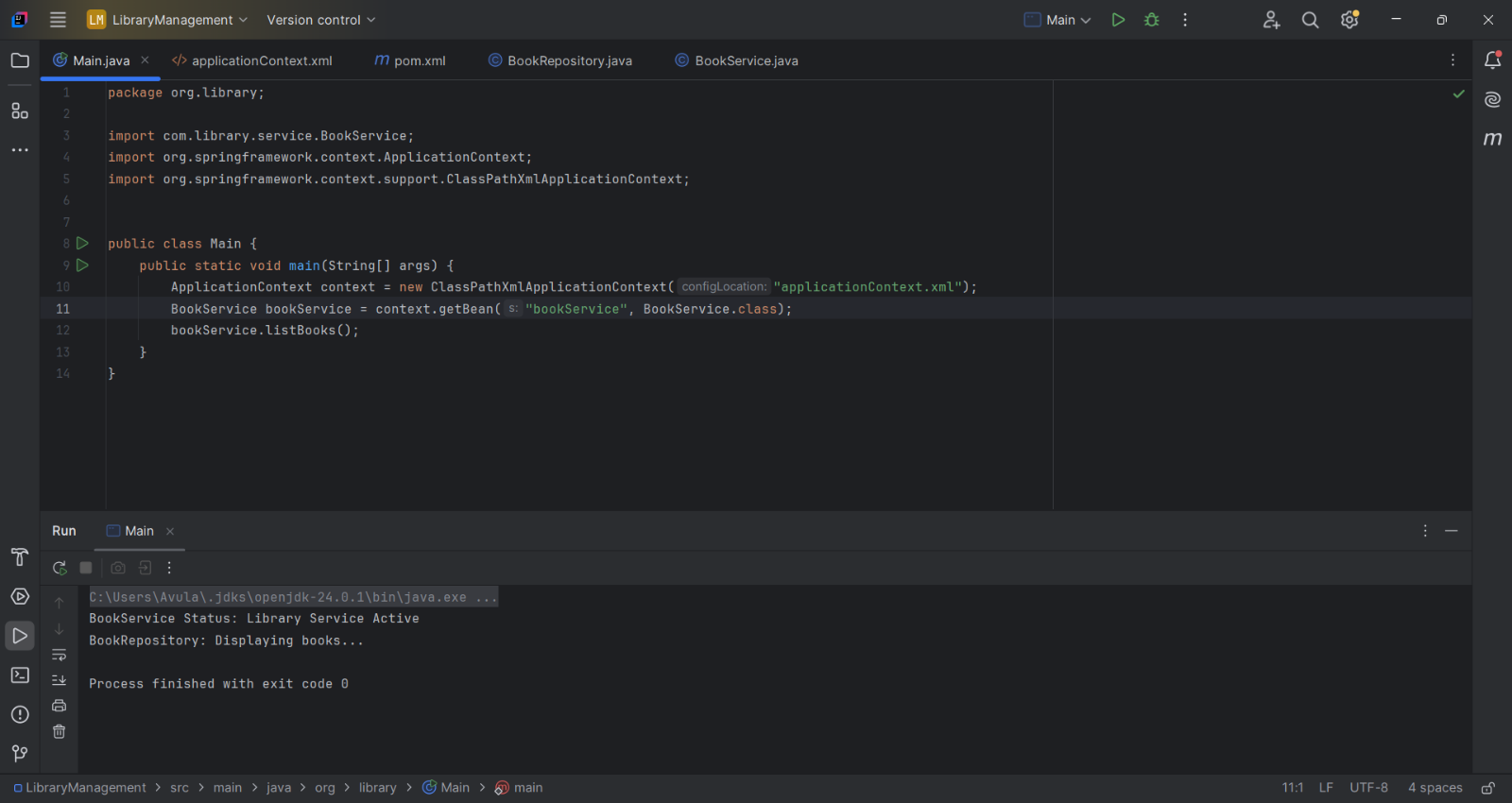
BookService bookService = context.getBean("bookService", BookService.class);

bookService.listBooks();

}

}

**Output**:



In this exercise, we implemented **dependency injection** using **XML configuration only**, without annotations, to configure and manage objects (beans) in a Spring-based **Library Management** application.

|  |  |
| --- | --- |
| **Concept** | **Description** |
| **IoC Container** | Central place where Spring manages object creation and dependencies |
| **Constructor Injection** | Used when a value is **mandatory** during object creation |
| **Setter Injection** | Used when a dependency can be set **after** object is created |
| **Loose Coupling** | Spring injects dependencies, avoiding tight coupling with new keyword |
| **XML Configuration** | Helps configure beans without touching source code (ideal for legacy apps) |