**Exercise 1: Configuring a Basic Spring Application**

Scenario:

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**Introduction**

This document outlines the steps taken to configure a basic Spring application for managing a library. The application consists of a backend that handles book-related operations, using the Spring Framework to manage dependencies and configurations.

**1. Setting Up the Spring Project**

**Project Creation**

A Maven project named **LibraryManagement** was created to manage the library system. Maven was chosen for dependency management and project structure.

**Dependencies**

* The Spring Core dependencies were added to the pom.xml file. These dependencies are essential for setting up the Spring Context and handling the beans within the application.

**2. Configuring the Application Context**

* An XML configuration file named applicationContext.xml was created in the src/main/resources directory. This file defines the beans for BookService and BookRepository. This configuration file establishes the relationships between the BookService and BookRepository classes, allowing Spring to manage their lifecycle.

**3. Service and Repository Classes**

**BookService**

The BookService class handles the business logic for managing books. It depends on BookRepository to perform operations like add books, print books and perform services.

**BookRepository**

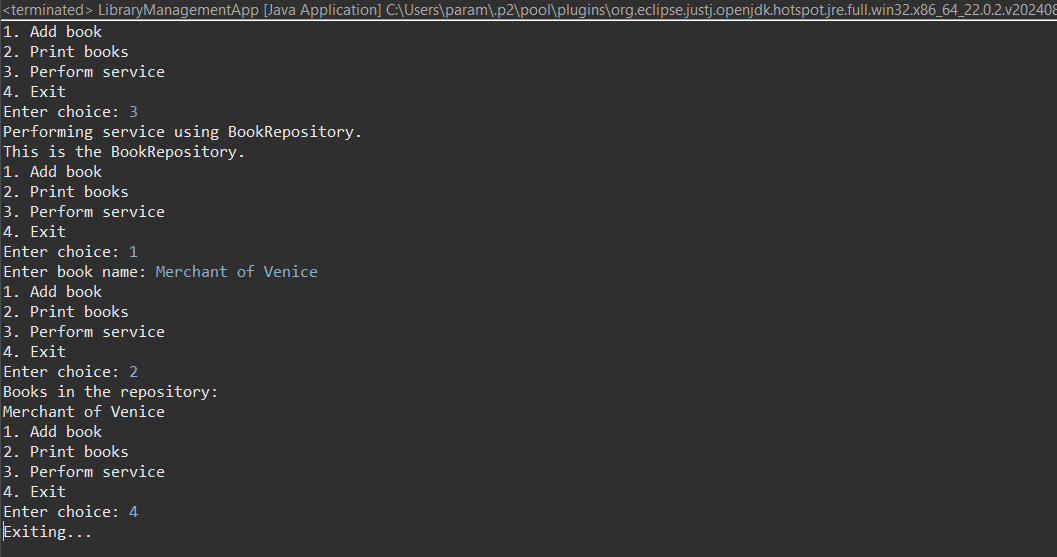
The BookRepository class manages a collection of Book objects, providing methods to add, retrieve, display repository information and print repository.

**4. Running the Application**

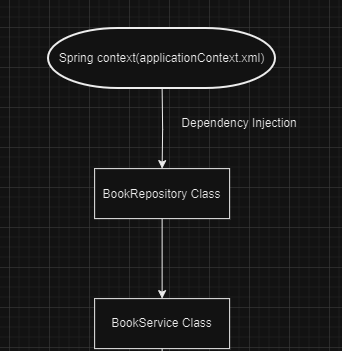
**Main Class**

A main class was created to load the Spring context and test the configuration. This class initializes the Spring framework and verifies that the BookService and BookRepository beans are correctly wired.

Output:



**FLOWCHART of the program :**



**Flowchart Explanation**

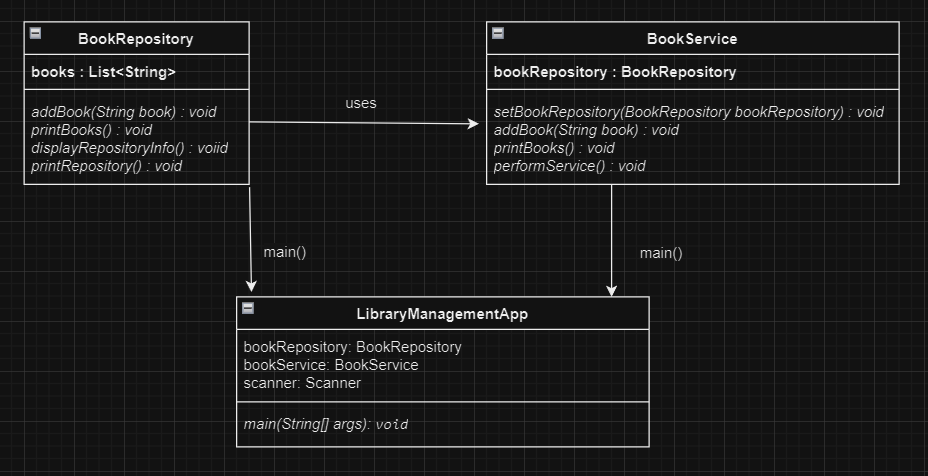
The flowchart represents the interactions between the key components of the Library Management application:

1. **Spring Context (applicationContext.xml)**:
   * This is where the Spring beans are defined and managed. It acts as the configuration center, handling the creation and injection of dependencies between different components.
2. **BookRepository**:
   * The BookRepository bean is created by Spring as per the configuration in applicationContext.xml. This class is responsible for managing the collection of Book objects. It provides methods to add new books, retrieve all books, display repository information and print repository.
3. **BookService**:
   * The BookService bean is also managed by Spring and depends on BookRepository. The Spring context injects BookRepository into BookService, allowing BookService to delegate book management tasks to BookRepository.

**Flow of Execution:**

* The Spring context initializes and wires the BookRepository and BookService beans.
* BookService interacts with BookRepository.
* All interactions are managed by Spring, ensuring that dependencies are correctly injected and managed throughout the application lifecycle.

**CLASS DIAGRAM :**



**Class Diagram Explanation**

The class diagram provides a static view of the system, showing the structure of the classes and their relationships:

1. **BookRepository Class**:
   * **Attributes**: books (a list of Book).
   * **Methods**:
     + addBook(String book): Adds a new book to the repository.
     + printBooks(): Prints all the books in the repository.
     + displayRepositoryinfo(): Provides information about the BookRepository.
     + printRepository(): Prints the BookRepository.
   * This class is responsible for managing and storing the books.
2. **BookService Class**:
   * **Attributes**: bookRepository (a reference to BookRepository).
   * **Methods**:
     + setBookRepository(BookRepository bookRepository): Injects the BookRepository dependency.
     + addBook(String book): Creates a new book and adds it to the repository.
     + printBooks(): Prints all books in the repository.
     + performService(): Services performed by the BookService class.
   * This class contains the business logic of the application. It uses BookRepository and is responsible for higher-level operations that may involve multiple books or more complex logic.
3. **LibraryManagementApp Class:**
   * **Attributes**:

* bookRepository: This attribute is an instance of the BookRepository class. It is used to store and manage the list of books.
* bookService: This attribute is an instance of the BookService class. It is used to perform operations related to books, such as adding a book and printing the list of books. It also has a setter method to set the BookRepository instance.
* scanner: This attribute is an instance of the Scanner class. It is used to read user input from the console.
  + **Methods**:
    - main(String[] args): main() method
* In the LibraryManagementApp class, the dependency is injected by creating an instance of BookRepository and setting it in the BookService instance using the setter method

**Relationships:**

* **Dependency (BookService to BookRepository)**: BookService relies on BookRepository to manage the actual data of the books.