**Library management application solution**

1. Set up the Project:

Create a new Spring Boot project with the required dependencies.

Create the following packages: **com.example.library.model**, **com.example.library.repository**, and **com.example.library.service**.

1. Create the Book Model:

package com.example.library.model;

public class Book {

private Long id;

private String title;

private String author;

private int publicationYear;

private int quantity;

// Constructors, getters, and setters

// Override toString() for easier printing

}

}

1. Create the Book Repository:

package com.example.library.repository;

import com.example.library.model.Book;

import org.springframework.stereotype.Repository;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

@Repository

public class BookRepository {

private Map<Long, Book> books = new ConcurrentHashMap<>();

public void addBook(Book book) {

books.put(book.getId(), book);

}

public void updateBook(Book book) {

books.put(book.getId(), book);

}

public void deleteBook(Long id) {

books.remove(id);

}

public Book getBookById(Long id) {

return books.get(id);

}

}

1. Create the Book Service:

package com.example.library.service;

import com.example.library.model.Book;

import com.example.library.repository.BookRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class BookServiceImpl implements BookService {

private final BookRepository bookRepository;

@Autowired

public BookServiceImpl(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

@Override

public void addBook(Book book) {

bookRepository.addBook(book);

}

@Override

public void updateBook(Book book) {

bookRepository.updateBook(book);

}

@Override

public void deleteBook(Long id) {

bookRepository.deleteBook(id);

}

@Override

public Book getBookById(Long id) {

return bookRepository.getBookById(id);

}

@Override

public void performInventoryCheck() {

bookRepository.getAllBooks().forEach(book -> {

if (book.getQuantity() < 5) {

System.out.println("Inventory Check: Book with ID " + book.getId() + " has low quantity.");

}

});

}

}

1. Configure Bean Lifecycle:

* No additional code is required for this step. The annotations **@Service**, **@PostConstruct**, **@Scheduled**, and **@PreDestroy** are already used in the code.

1. Test the Application:

package com.example.library;

import com.example.library.model.Book;

import com.example.library.service.BookService;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

@SpringBootApplication

public class LibraryApplication {

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(LibraryApplication.class, args);

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

// Add sample books

bookService.addBook(new Book(1L, "To Kill a Mockingbird", "Harper Lee", 1960, 10));

bookService.addBook(new Book(2L, "1984", "George Orwell", 1949, 8));

bookService.addBook(new Book(3L, "The Great Gatsby", "F. Scott Fitzgerald", 1925, 15));

bookService.addBook(new Book(4L, "The Catcher in the Rye", "J.D. Salinger", 1951, 12));

bookService.addBook(new Book(5L, "The Hobbit", "J.R.R. Tolkien", 1937, 7));

bookService.addBook(new Book(6L, "Moby-Dick", "Herman Melville", 1851, 6));

bookService.addBook(new Book(7L, "Pride and Prejudice", "Jane Austen", 1813, 20));

bookService.addBook(new Book(8L, "The Lord of the Rings", "J.R.R. Tolkien", 1954, 14));

bookService.addBook(new Book(9L, "War and Peace", "Leo Tolstoy", 1869, 5));

bookService.addBook(new Book(10L, "The Da Vinci Code", "Dan Brown", 2003, 10));

// Perform inventory check

bookService.performInventoryCheck();

}

}