**Assignment Topic:** Library Management System - Inventory Management

**Objective:** Develop a simplified Library Management System using Spring Boot, concentrating on Core concepts, Autowiring, and Bean Lifecycle. The system will manage an inventory of books and perform periodic checks on the inventory.

**Task Details:**

1. **Setting up the Project** Set up a new Spring Boot project and include the necessary dependencies. Your project should have the following packages:
   * model
   * repository
   * service
2. **Creating the Model** Create a **Book** class in the model package. The **Book** class should have the following fields:
   * id (Long)
   * title (String)
   * author (String)
   * publicationYear (int)
   * quantity (int)
3. **Creating the Repository** Create a **BookRepository** class in the repository package. This will act as a mock database for this assignment. It should have methods to add, update, delete, and get books. Use a **Map<Long, Book>** to store the books, where the key is the book id and the value is the book itself.
4. **Creating the Service** Create a **BookService** interface and a **BookServiceImpl** class in the service package.

In the **BookServiceImpl** class, use the **@Autowired** annotation to inject the **BookRepository**. This is where the Autowiring comes into play. Spring will automatically take care of wiring the **BookRepository** into the **BookServiceImpl**, reducing the need for manual configuration and making the code cleaner and easier to maintain.

Implement methods to add, update, delete, and get books in the **BookServiceImpl** class. These methods should call the corresponding methods in **BookRepository**.

1. **Implementing the Inventory Check** In the **BookServiceImpl** class, add a method that checks the quantity of each book in the inventory and prints a message if the quantity is below a certain threshold. This method will simulate a periodic inventory check.
2. **Understanding the Bean Lifecycle** Use the **@PostConstruct** annotation to call the inventory check method when the **BookServiceImpl** bean is initialized. This will simulate an initial inventory check when the system starts up.

Use the **@Scheduled** annotation to call the inventory check method at regular intervals. This will simulate periodic inventory checks while the system is running.

Use the **@PreDestroy** annotation to print a message when the **BookServiceImpl** bean is destroyed. This will simulate a final inventory check when the system shuts down.

**Submission:** Submit the entire project folder as a ZIP file. Make sure to include any setup instructions and dependencies in the README.md file.

**Evaluation Criteria:**

1. Correct setup and organization of the Spring Boot project.
2. Proper use of Autowiring to manage dependencies between **BookServiceImpl** and **BookRepository**.
3. Correct implementation of the service methods in the **BookServiceImpl**.
4. Correct implementation of the repository methods in the **BookRepository**.
5. Understanding of the Bean Lifecycle demonstrated by the use of **@PostConstruct**, **@Scheduled** and **@PreDestroy**.

**Sample logic to create book objects**

@PostConstruct

public void initBooks() {

Book book1 = new Book(1L, "To Kill a Mockingbird", "Harper Lee", 1960, 10);

Book book2 = new Book(2L, "1984", "George Orwell", 1949, 8);

Book book3 = new Book(3L, "The Great Gatsby", "F. Scott Fitzgerald", 1925, 15);

Book book4 = new Book(4L, "The Catcher in the Rye", "J.D. Salinger", 1951, 12);

Book book5 = new Book(5L, "The Hobbit", "J.R.R. Tolkien", 1937, 7);

Book book6 = new Book(6L, "Moby-Dick", "Herman Melville", 1851, 6);

Book book7 = new Book(7L, "Pride and Prejudice", "Jane Austen", 1813, 20);

Book book8 = new Book(8L, "The Lord of the Rings", "J.R.R. Tolkien", 1954, 14);

Book book9 = new Book(9L, "War and Peace", "Leo Tolstoy", 1869, 5);

Book book10 = new Book(10L, "The Da Vinci Code", "Dan Brown", 2003, 10);

bookService.addBook(book1);

bookService.addBook(book2);

bookService.addBook(book3);

bookService.addBook(book4);

bookService.addBook(book5);

bookService.addBook(book6);

bookService.addBook(book7);

bookService.addBook(book8);

bookService.addBook(book9);

bookService.addBook(book10);

}

Or use addAll method to all all books at once.