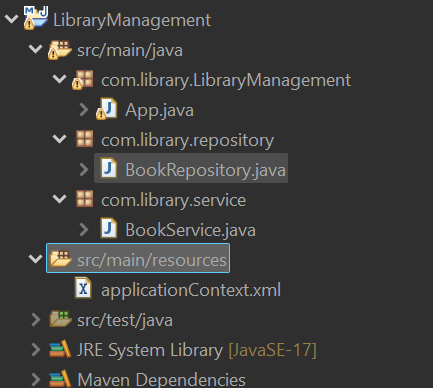
**Spring Core and Maven**

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

1. **Created a Maven project named LibraryManagement.**
2. **Added Spring Core dependencies in the pom.xml file.**
3. **Created an XML configuration file named applicationContext.xml in the src/main/resources directory.**

****

1. **Define beans for BookService and BookRepository in the XML file.**

**Code:**

<?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="https://www.springframework.org/schema/beans

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="bookService" class="com.library.service.BookService"/>

<bean id="bookRepository" class="com.library.repository.BookRepository"/>

</beans>

1. **Created a package com.library.service and add a class BookService.**

**Code:**

package com.library.service;

public class BookService {

public void addBook(String name) {

System.out.println("Book added: " + name);

}

}

1. **Created a package com.library.repository and add a class BookRepository.**

**Code:**

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.*out*.println("Book saved to database: " + bookName);

}

}

1. **Created a main class to load the Spring context and test the configuration.**

**Code:**

package com.library.LibraryManagement;

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

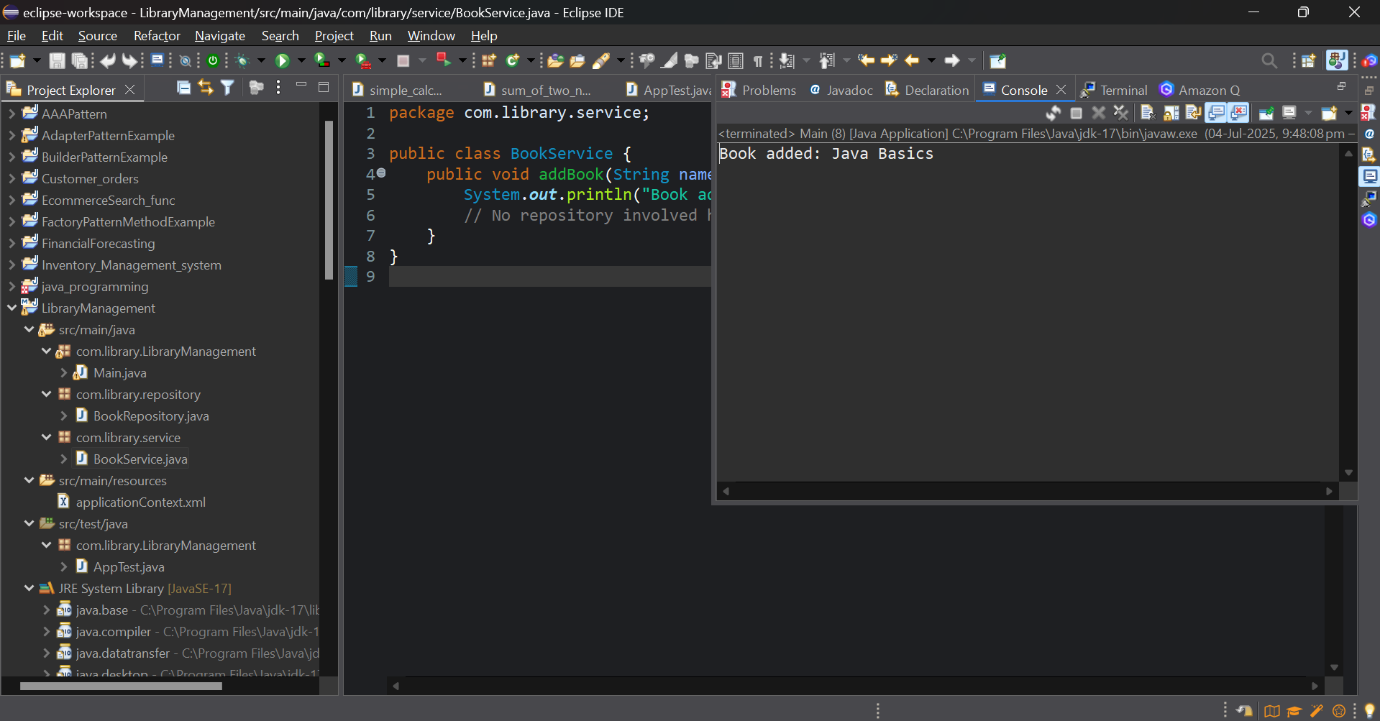
service.addBook("Java Basics");

}

}

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 2: Implementing Dependency Injection**

**1. Updated applicationContext.xml to wire BookRepository into BookService.**

**Code:**

<?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="https://www.springframework.org/schema/beans

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="bookRepository" class="com.library.repository.BookRepository"/>

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository"/>

</bean>

</beans>

**2. Updated the BookService Class:**

**Code:**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.out.println("Adding book: " + bookName);

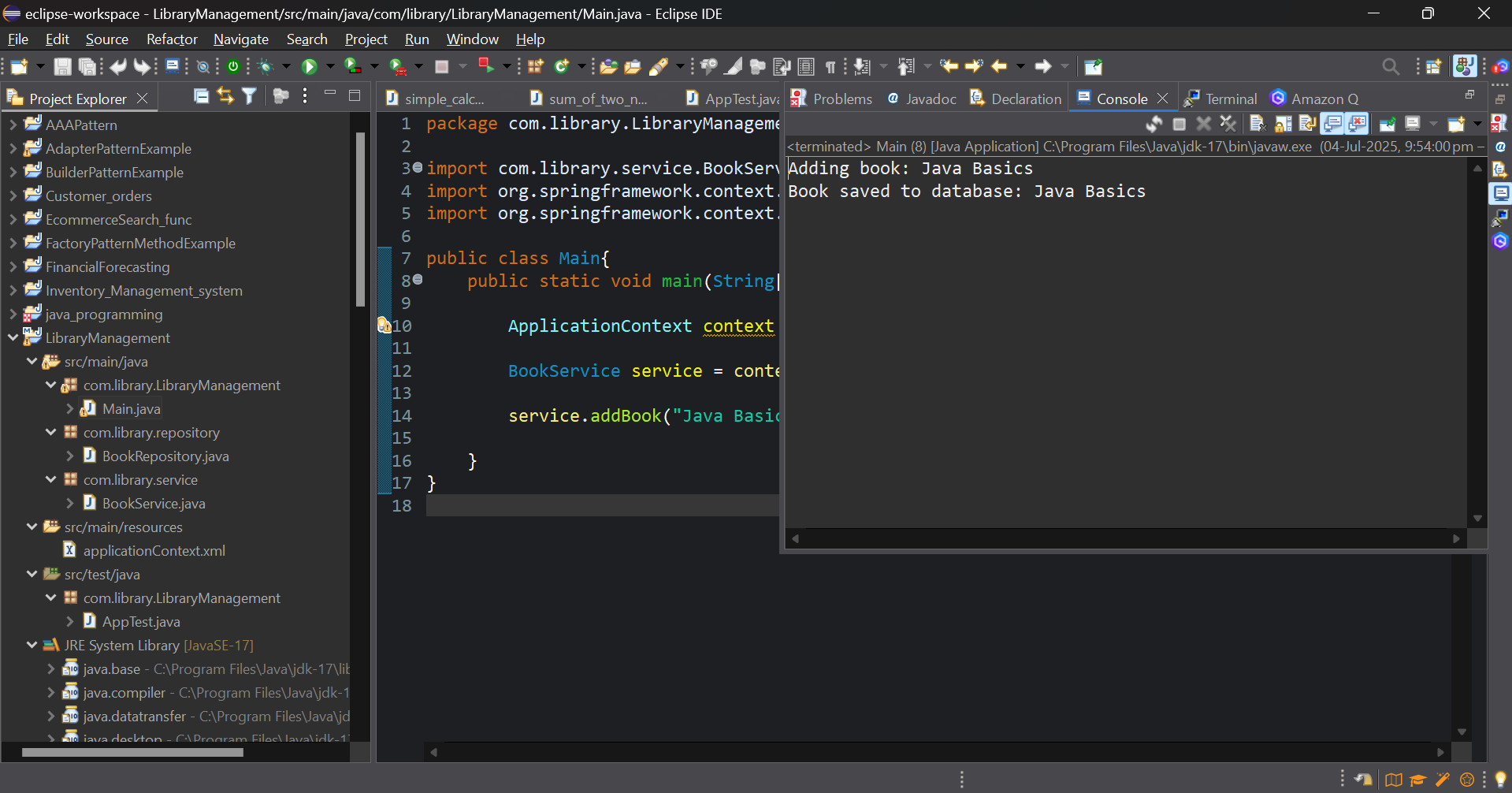
bookRepository.saveBook(bookName);

}

}

**3. Ran the LibraryManagementApplication main class to verify the dependency injection.**

**Output:**



A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 3: Implementing Logging with Spring AOP**

1. **Added Spring AOP Dependency:**
2. **Created a package com.library.aspect and add a class LoggingAspect with a method to log execution times.**

**Code:**

package com.library.aspect;

import org.aspectj.lang.ProceedingJoinPoint;

public class LoggingAspect{

public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {

long start = System.*currentTimeMillis*();

Object returnValue = joinPoint.proceed();

long end = System.*currentTimeMillis*();

System.*out*.println( joinPoint.getSignature().getName() +

" executed in " + (end - start) + " ms");

return returnValue;

}

}

1. **Updated applicationContext.xml to enable AspectJ support and registered the aspect.**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="

http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/aop https://www.springframework.org/schema/aop/spring-aop.xsd">

<aop:aspectj-autoproxy/>

<bean id="bookRepository" class="com.library.repository.BookRepository"/>

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository"/>

</bean>

<bean id="loggingAspect" class="com.library.aspect.LoggingAspect"/>

<aop:config>

<aop:aspect ref="loggingAspect">

<!-- Around advice: apply to all methods in BookService -->

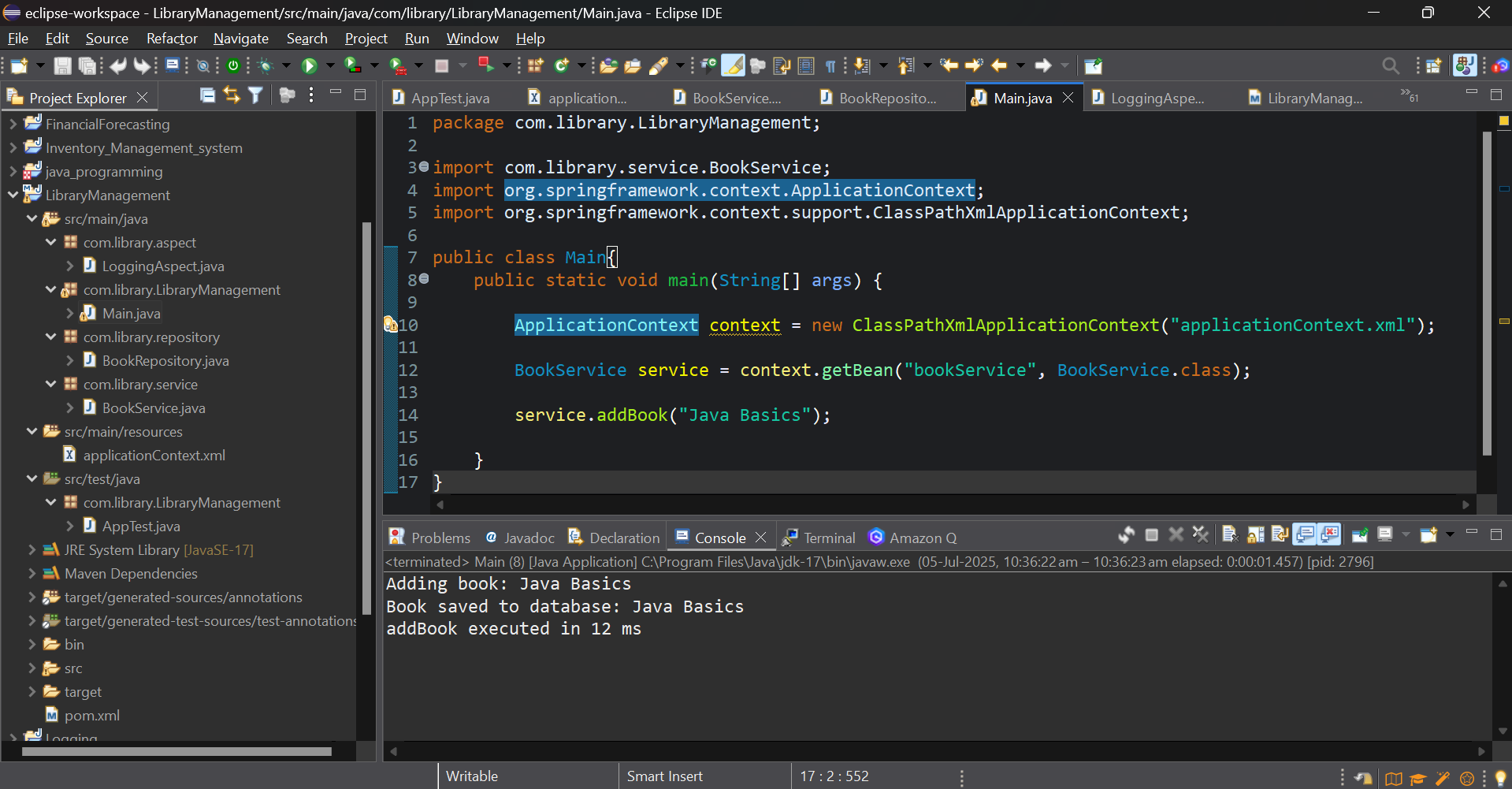
<aop:around method="logExecutionTime"

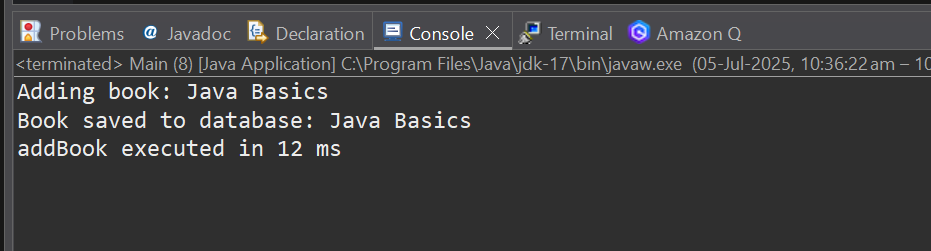
pointcut="execution(\* com.library.service.\*.\*(..))"/>

</aop:aspect>

</aop:config>

</beans>

1. **Ouput : of the console with log messages indicating method execution times.** ****



**Exercise 4: Creating and Configuring a Maven Project**

1. **Created a new Maven project named LibraryManagement.**
2. **Included dependencies for Spring Context, Spring AOP, and Spring WebMVC.**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring AOP (if needed later) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring WebMVC (needed if using controllers, JSP, etc.) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

1. **Configured the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.**

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

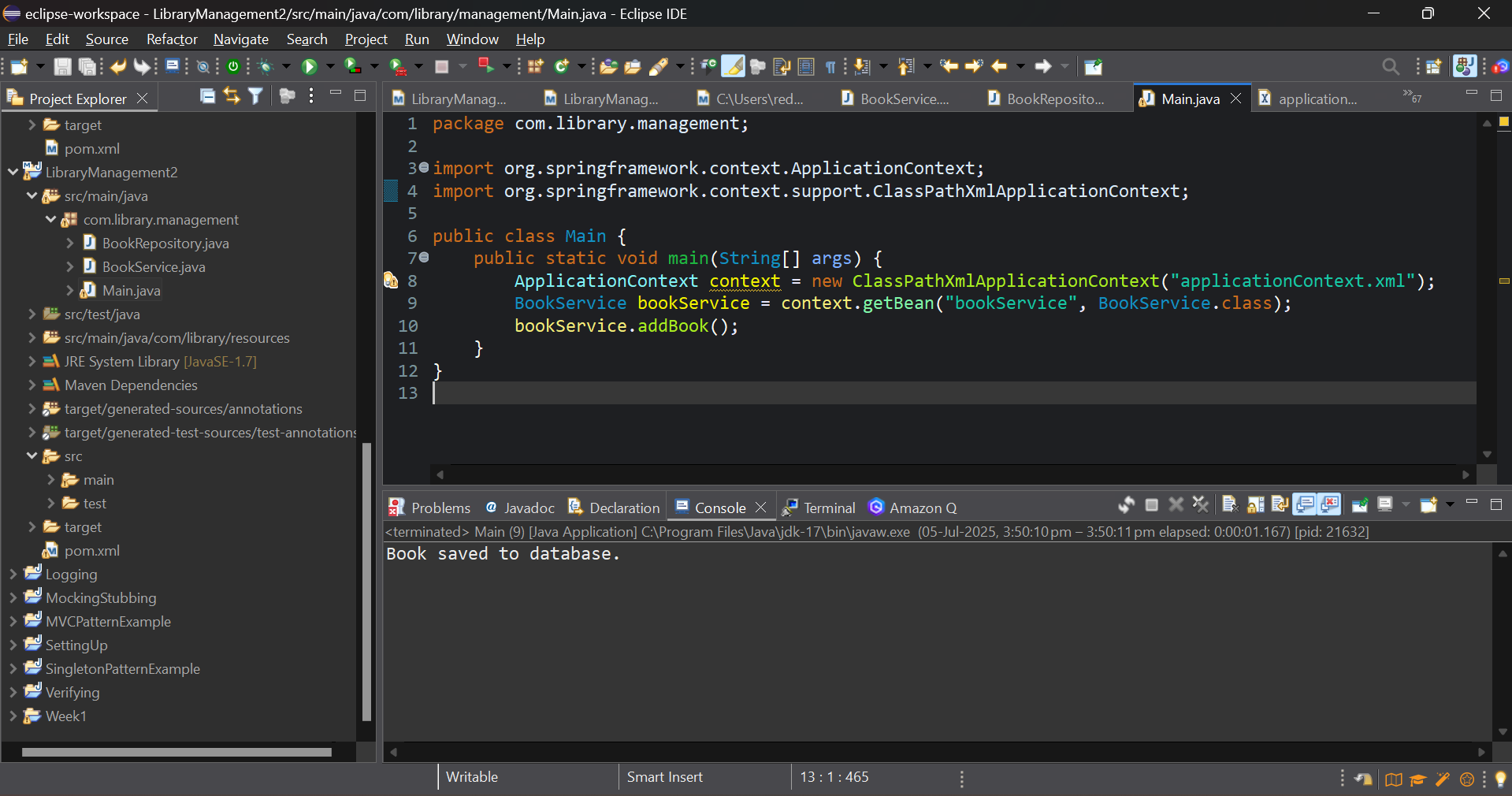
<source>1.8</source>

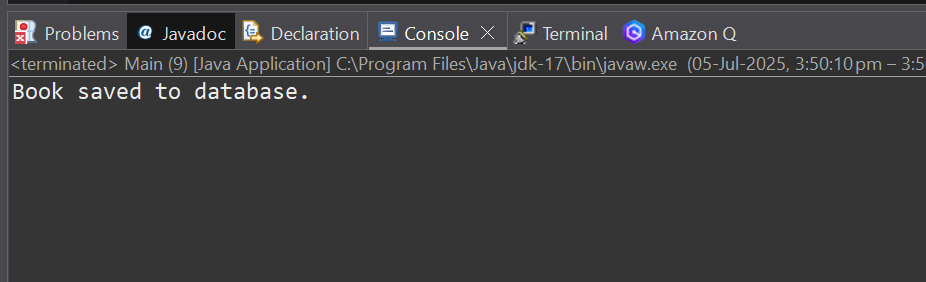
<target>1.8</target>

</configuration>

</plugin>

**Output:**

****

****