**Exercise 2: Implementing Dependency Injection**

**Book.java**

package com.library.model;

public class Book {

private int id;

private String title;

private String author;

private boolean available = true;

public Book() {}

public Book(int id, String title, String author) {

this.id = id;

this.title = title;

this.author = author;

}

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public String getAuthor() { return author; }

public void setAuthor(String author) { this.author = author; }

public boolean isAvailable() { return available; }

public void setAvailable(boolean available) { this.available = available; }

@Override

public String toString() {

return "[" + id + "] " + title + " by " + author + (available ? " (Available)" : " (Not Available)");

}

}

**BookRepository.java**

package com.library.repository;

import com.library.model.Book;

import java.util.ArrayList;

import java.util.List;

public interface BookRepository {

List<Book> findAll();

}

**BookRepositoryImpl.java**

package com.library.repository;

import com.library.model.Book;

import java.util.ArrayList;

import java.util.List;

public class BookRepositoryImpl implements BookRepository {

private final List<Book> books = new ArrayList<>();

public BookRepositoryImpl() {

books.add(new Book(1, "Effective Java", "Joshua Bloch"));

books.add(new Book(2, "Clean Code", "Robert C. Martin"));

books.add(new Book(3, "Spring in Action", "Craig Walls"));

}

@Override

public List<Book> findAll() {

return books;

}

}

**BookService.java**

package com.library.service;

public interface BookService {

void printAvailableBooks();

}

# **BookServiceImpl.java**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

import java.util.List;

public class BookServiceImpl implements BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

@Override

public void printAvailableBooks() {

List<Book> books = bookRepository.findAll();

System.out.println("===== Available Books =====");

for (Book book : books) {

System.out.println(book);

}

}

}

# **BookServiceImplTest.java**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

import org.junit.Before;

import org.junit.Test;

import java.util.Arrays;

import java.util.List;

import static org.mockito.Mockito.\*;

public class BookServiceImplTest {

private BookRepository bookRepository;

private BookServiceImpl bookService;

@Before

public void setUp() {

bookRepository = mock(BookRepository.class);

bookService = new BookServiceImpl();

bookService.setBookRepository(bookRepository);

}

@Test

public void testPrintAvailableBooks() {

List<Book> mockBooks = Arrays.asList(

new Book(1, "TDD", "Kent Beck"),

new Book(2, "Refactoring", "Martin Fowler")

);

when(bookRepository.findAll()).thenReturn(mockBooks);

bookService.printAvailableBooks();

verify(bookRepository, times(1)).findAll();

}

}

**BorrowingServiceImplTest.java**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

import org.junit.Before;

import org.junit.Test;

import java.util.Arrays;

import static org.mockito.Mockito.\*;

import com.library.service.BorrowingServiceImpl;

public class BorrowingServiceImplTest {

private BorrowingServiceImpl borrowingService;

private BookRepository bookRepository;

@Before

public void setUp() {

bookRepository = mock(BookRepository.class);

borrowingService = new BorrowingServiceImpl();

borrowingService.setBookRepository(bookRepository);

}

@Test

public void testBorrowAvailableBook() {

Book book = new Book(1, "Domain-Driven Design", "Eric Evans");

book.setAvailable(true);

when(bookRepository.findAll()).thenReturn(Arrays.asList(book));

borrowingService.borrowBookById(1);

}

@Test

public void testBorrowAlreadyBorrowedBook() {

Book book = new Book(2, "PoEAA", "Martin Fowler");

book.setAvailable(false);

when(bookRepository.findAll()).thenReturn(Arrays.asList(book));

borrowingService.borrowBookById(2);

}

@Test

public void testBorrowNonExistentBook() {

when(bookRepository.findAll()).thenReturn(Arrays.asList());

borrowingService.borrowBookById(999);

}

}

**LibraryManagementApplication.java**

package com.library.main;

import com.library.service.BookService;

import com.library.service.BorrowingService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {

public static void main(String[] args) {

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

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

BorrowingService borrowingService = context.getBean("borrowingService", BorrowingService.class);

bookService.printAvailableBooks();

borrowingService.borrowBookById(2);

borrowingService.borrowBookById(2);

borrowingService.borrowBookById(99);

bookService.printAvailableBooks();

}

}

**Library\_Management\_System.iml**

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

<module version="4">

<component name="AdditionalModuleElements">

<content url="file://$MODULE\_DIR$" dumb="true">

<sourceFolder url="file://$MODULE\_DIR$/src/main/java" isTestSource="false" />

<sourceFolder url="file://$MODULE\_DIR$/src/test/java" isTestSource="true" />

</content>

</component>

</module>

**Pom.xml**

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

<project xmlns="http://maven.apache.org/POM/4.0.0"

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.example</groupId>

<artifactId>Library\_Management\_System</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>22</maven.compiler.source>

<maven.compiler.target>22</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.11.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.hamcrest</groupId>

<artifactId>hamcrest</artifactId>

<version>2.2</version>

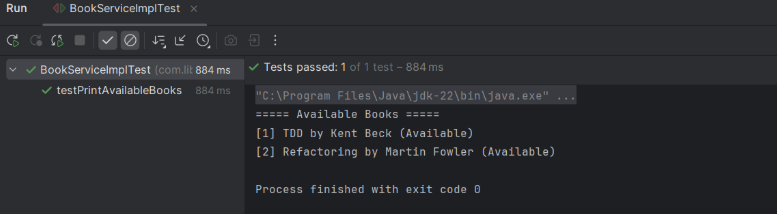
<scope>test</scope>

</dependency>

</dependencies>

</project>

**OUTPUT :**

****