**Exercise 7: Implementing Constructor and Setter Injection**

**Book.java**

package com.library.model;

import java.time.LocalDate;

public class Book {

private int id;

private String title;

private String author;

private boolean available;

private LocalDate borrowedDate;

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

this.id = id;

this.title = title;

this.author = author;

this.available = available;

}

public int getId() { return id; }

public String getTitle() { return title; }

public String getAuthor() { return author; }

public boolean isAvailable() { return available; }

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

public LocalDate getBorrowedDate() { return borrowedDate; }

public void setBorrowedDate(LocalDate borrowedDate) { this.borrowedDate = borrowedDate; }

@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.List;

public interface BookRepository {

List<Book> findAll();

Book findById(int id);

void update(Book book);

List<Book> findBorrowedBooks();

}

**BookRepositoryImpl.java**

package com.library.repository;

import com.library.model.Book;

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

public class BookRepositoryImpl implements BookRepository {

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

public BookRepositoryImpl() {

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

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

books.add(new Book(3, "Refactoring", "Martin Fowler", false));

}

@Override

public List<Book> findAll() {

return books;

}

@Override

public Book findById(int id) {

return books.stream().filter(b -> b.getId() == id).findFirst().orElse(null);

}

@Override

public void update(Book book) {

for (int i = 0; i < books.size(); i++) {

if (books.get(i).getId() == book.getId()) {

books.set(i, book);

break;

}

}

}

@Override

public List<Book> findBorrowedBooks() {

return books.stream().filter(b -> !b.isAvailable()).collect(Collectors.toList());

}

}

**BookServiceImpl.java**

package com.library.service;

import com.library.model.Book;

import com.library.repository.BookRepository;

import java.util.List;

public class BookServiceImpl {

private BookRepository bookRepository;

public BookServiceImpl(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBooks() {

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

for (Book book : books) {

System.out.println(book);

}

}

public void printAvailableBooks() {

bookRepository.findAll().stream()

.filter(Book::isAvailable)

.forEach(book -> System.out.println("Available: " + 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(bookRepository);

}

@Test

public void testDisplayBooks() {

List<Book> books = Arrays.asList(

new Book(1, "Effective Java", "Joshua Bloch", true),

new Book(2, "Clean Code", "Robert C. Martin", false)

);

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

bookService.displayBooks();

}

}

**Main.java**

package com.library.main;

import com.library.repository.BookRepositoryImpl;

import com.library.service.BookServiceImpl;

import com.library.service.BorrowingServiceImpl;

public class Main {

public static void main(String[] args) throws InterruptedException {

BookRepositoryImpl repo = new BookRepositoryImpl();

BookServiceImpl bookService = new BookServiceImpl(repo); // constructor injection

bookService.setBookRepository(repo); // setter injection

BorrowingServiceImpl borrowService = new BorrowingServiceImpl();

borrowService.setBookRepository(repo);

System.out.println("All Books:");

bookService.displayBooks();

System.out.println("\nBorrow operations:");

borrowService.borrowBookById(2);

borrowService.borrowBookById(3);

System.out.println("\nBorrowed Books:");

borrowService.displayBorrowedBooks();

System.out.println("\nReturning Book:");

borrowService.returnBookWithFine(2);

System.out.println("\nAvailable Books:");

bookService.printAvailableBooks();

}

}

# **applicationContext.xml**

<?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="

http://www.springframework.org/schema/beans

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

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

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

<constructor-arg ref="bookRepository" />

<property name="bookRepository" ref="bookRepository"/> <!-- Setter Injection -->

</bean>

<bean id="borrowingService" class="com.library.service.BorrowingServiceImpl">

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

</bean>

</beans>

**Pom.xml**

<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>Setter\_Injection</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.hamcrest</groupId>

<artifactId>hamcrest</artifactId>

<version>2.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<version>3.11.0</version>

<configuration>

<source>22</source>

<target>22</target>

</configuration>

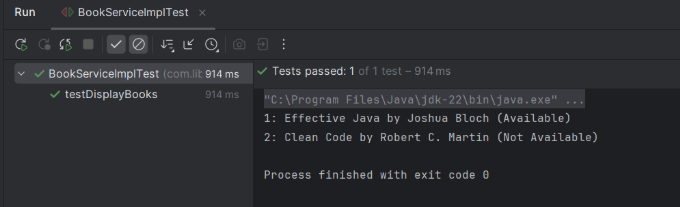
</plugin>

</plugins>

</build>

</project>

**OUTPUT :**

****