**Exercise 2: Verifying Interactions**

package com.hubberspot.mockito.behavior.verfication;

import com.hubberspot.mockito.behavior.verification.Book;

import com.hubberspot.mockito.behavior.verification.BookRepository;

import com.hubberspot.mockito.behavior.verification.BookRequest;

import com.hubberspot.mockito.behavior.verification.BookService;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.\*;

import org.mockito.junit.jupiter.MockitoExtension;

import java.time.LocalDate;

import static org.mockito.Mockito.\*;

@ExtendWith(MockitoExtension.class)

public class BookServiceTest {

@InjectMocks

private BookService bookService;

// @Mock

// private BookRepository bookRepository;

@Spy

private BookRepository bookRepository;

@Test

public void testAddBook() {

Book book = new Book(null, "Mockito In Action", 600, LocalDate.now());

bookService.addBook(book);

verify(bookRepository).save(book);

}

@Test

public void testSaveBookWithBookRequestWithGreaterPrice() {

BookRequest bookRequest = new BookRequest("Mockito In Action", 500, LocalDate.now());

Book book = new Book(null, "Mockito In Action", 500, LocalDate.now());

bookService.addBook(bookRequest);

verify(bookRepository, times(0)).save(book);

}

@Test

public void testSaveBookWithBookRequestWithGreaterPrice1() {

BookRequest bookRequest = new BookRequest("Mockito In Action", 600, LocalDate.now());

Book book = new Book(null, "Mockito In Action", 600, LocalDate.now());

bookService.addBook(bookRequest);

bookService.addBook(bookRequest);

verify(bookRepository, times(2)).save(book);

}

@Test

public void testSaveBookWithBookRequestWithGreaterPrice2() {

BookRequest bookRequest = new BookRequest("Mockito In Action", 500, LocalDate.now());

Book book = new Book(null, "Mockito In Action", 500, LocalDate.now());

bookService.addBook(bookRequest);

verify(bookRepository, never()).save(book);

}

@Test

public void testUpdatePrice() {

bookService.updatePrice(null, 600);

verifyNoInteractions(bookRepository);

}

@Test

public void testUpdatePrice2() {

Book book = new Book("1234", "Mockito In Action", 500, LocalDate.now());

when(bookRepository.findBookById("1234")).thenReturn(book);

bookService.updatePrice("1234", 500);

verify(bookRepository).findBookById("1234");

verify(bookRepository).save(book);

verifyNoMoreInteractions(bookRepository);

}

@Test

public void testUpdatePrice3() {

Book book = new Book("1234", "Mockito In Action", 500, LocalDate.now());

when(bookRepository.findBookById("1234")).thenReturn(book);

bookService.updatePrice("1234", 500);

InOrder inOrder = Mockito.inOrder(bookRepository);

inOrder.verify(bookRepository).findBookById("1234");

inOrder.verify(bookRepository).save(book);

}

@Test

public void testSaveBookWithBookRequestWithGreaterPrice3() {

BookRequest bookRequest = new BookRequest("Mockito In Action", 600, LocalDate.now());

Book book = new Book(null, "Mockito In Action", 600, LocalDate.now());

bookService.addBook(bookRequest);

bookService.addBook(bookRequest);

bookService.addBook(bookRequest);

// verify(bookRepository, times(2)).save(book);

// verify(bookRepository, atLeast(4)).save(book);

// verify(bookRepository, atMost(2)).save(book);

// verify(bookRepository, atMostOnce()).save(book);

verify(bookRepository, atLeastOnce()).save(book);

}

}

Output :

