**Mocking and Stubbing with Mockito**

**1. Scenario Setup**

We'll test a FlightBookingService that depends on an external PaymentGateway service.

**1.1 Create the External Dependency Interface**

public interface PaymentGateway {

boolean processPayment(double amount, String cardNumber);

String getLastTransactionId();

boolean isAvailable();

}

**1.2 Create the Service to Test**

java

Copy

Download

public class FlightBookingService {

private final PaymentGateway paymentGateway;

public FlightBookingService(PaymentGateway paymentGateway) {

this.paymentGateway = paymentGateway;

}

public BookingStatus bookFlight(String flightId, String passengerId,

String cardNumber, double amount) {

if (!paymentGateway.isAvailable()) {

return BookingStatus.PAYMENT\_SYSTEM\_UNAVAILABLE;

}

boolean paymentSuccess = paymentGateway.processPayment(amount, cardNumber);

if (paymentSuccess) {

String confirmationCode = "BOOK-" + paymentGateway.getLastTransactionId();

return new BookingStatus(confirmationCode, true);

} else {

return BookingStatus.PAYMENT\_FAILED;

}

}

}

public enum BookingStatus {

SUCCESS(true, "Success"),

PAYMENT\_FAILED(false, "Payment failed"),

PAYMENT\_SYSTEM\_UNAVAILABLE(false, "Payment system unavailable");

// ... enum implementation

}

**2. Test Implementation with Mockito**

import org.junit.jupiter.api.Test;

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

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.junit.jupiter.MockitoExtension;

import static org.mockito.Mockito.\*;

import static org.junit.jupiter.api.Assertions.\*;

@ExtendWith(MockitoExtension.class)

public class FlightBookingServiceTest {

@Mock

private PaymentGateway mockPaymentGateway;

@InjectMocks

private FlightBookingService bookingService;

@Test

public void testSuccessfulBooking() {

// Arrange (Stubbing)

when(mockPaymentGateway.isAvailable()).thenReturn(true);

when(mockPaymentGateway.processPayment(anyDouble(), anyString())).thenReturn(true);

when(mockPaymentGateway.getLastTransactionId()).thenReturn("TX123456");

// Act

BookingStatus result = bookingService.bookFlight(

"FL123", "PASS001", "4111111111111111", 250.00);

// Assert

assertTrue(result.isSuccess());

assertEquals("BOOK-TX123456", result.getConfirmationCode());

// Verify interactions

verify(mockPaymentGateway).isAvailable();

verify(mockPaymentGateway).processPayment(250.00, "4111111111111111");

verify(mockPaymentGateway).getLastTransactionId();

}

@Test

public void testPaymentSystemUnavailable() {

// Arrange

when(mockPaymentGateway.isAvailable()).thenReturn(false);

// Act

BookingStatus result = bookingService.bookFlight(

"FL123", "PASS001", "4111111111111111", 250.00);

// Assert

assertEquals(BookingStatus.PAYMENT\_SYSTEM\_UNAVAILABLE, result);

verify(mockPaymentGateway, never()).processPayment(anyDouble(), anyString());

}

@Test

public void testPaymentFailure() {

// Arrange

when(mockPaymentGateway.isAvailable()).thenReturn(true);

when(mockPaymentGateway.processPayment(anyDouble(), anyString())).thenReturn(false);

// Act

BookingStatus result = bookingService.bookFlight(

"FL123", "PASS001", "4111111111111111", 250.00);

// Assert

assertEquals(BookingStatus.PAYMENT\_FAILED, result);

}

}

**Key Mockito Concepts Explained**

**1. Mock Creation**

* @Mock creates a mock implementation of PaymentGateway
* @InjectMocks creates the real service and injects the mock dependency

**2. Stubbing Methods**

when(mock.method(args)).thenReturn(value);

* Defines mock behavior for specific inputs
* Can use argument matchers like anyString(), anyDouble()

**3. Verification**

java

Copy

Download

verify(mock).method(args);

verify(mock, times(n)).method(args);

verify(mock, never()).method(args);

* Ensures expected interactions occurred
* Validates call counts

**4. Advanced Features**

java

Copy

Download

// Throw exceptions

when(mock.method()).thenThrow(new RuntimeException());

// Return different values on subsequent calls

when(mock.method())

.thenReturn(true)

.thenReturn(false);

// Verify call order

InOrder inOrder = inOrder(mock1, mock2);

inOrder.verify(mock1).method();

inOrder.verify(mock2).method();