Test driven development and Logging framework

SuperSet ID:6412063

Mockito

Exercise 7: Handling Void Methods with Exceptions

pom.xml

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.11.0</version>

<scope>test</scope>

</dependency>

</dependencies>

LoggerService.java

package com.example.service;

public interface LoggerService {

void log(String message) throws Exception; // This method throws an exception

}

UserProcessor.java

package com.example.service;

public class UserProcessor {

private LoggerService logger;

public UserProcessor(LoggerService logger) {

this.logger = logger;

}

public void processUser(String username) {

try {

logger.log("Processing user: " + username);

System.out.println("✅ User processed successfully.");

} catch (Exception e) {

System.out.println("❌ Exception caught while processing user: " + e.getMessage());

}

}

}

UserProcessorTest.java

package com.example.service;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class UserProcessorTest {

@Test

public void testVoidMethodThrowsException() throws Exception {

LoggerService mockLogger = mock(LoggerService.class);

doThrow(new RuntimeException("Logger failure")).when(mockLogger).log(anyString());

UserProcessor processor = new UserProcessor(mockLogger);

processor.processUser("dharshini123");

verify(mockLogger).log("Processing user: dharshini123");

System.out.println("✅ Test Completed: Exception behavior verified.");

}

}

Output:

A screenshot of a computer

Description automatically generated

Console output:

A black square with white text

Description automatically generated