Test driven development and Logging framework

SuperSet ID:6412063

Mockito

Exercise 4: Handling Void Methods

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); // void method

}

UserManager.java

package com.example.service;

public class UserManager {

private LoggerService logger;

public UserManager(LoggerService logger) {

this.logger = logger;

}

public void createUser(String username) {

System.out.println("Creating user: " + username); // console output

logger.log("User created: " + username);

}

}

UserManagerTest.java

package com.example.service;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class UserManagerTest {

@Test

public void testVoidMethodLogging() {

LoggerService mockLogger = Mockito.mock(LoggerService.class);

doNothing().when(mockLogger).log(anyString());

UserManager manager = new UserManager(mockLogger);

manager.createUser("dharshini123");

verify(mockLogger).log("User created: dharshini123");

System.out.println("✅ Test Passed: log() was called with expected message.");

}

}

Output:

A screenshot of a computer

Description automatically generated

Console Output:

A black screen with red text

Description automatically generated