**Mockito Exercises**

**Exercise 1: Mocking and Stubbing**

**Scenario:** Maven configuration file that includes dependencies for JUnit and Mockito.

**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 <https://maven.apache.org/xsd/maven-4.0.0.xsd>">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>mockito-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.10.0</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**UserDao.java**

package com.example.service;

public interface UserDao {

String getUserEmail(int userId);

}

**UserService.java**

package com.example.service;

public class UserService {

private UserDao userDao;

public UserService(UserDao userDao) {

this.userDao = userDao;

}

public String getUserEmailUpperCase(int userId) {

String email = userDao.getUserEmail(userId);

return (email != null) ? email.toUpperCase() : null;

}

}

**UserServiceTest.java**

package com.example.service;

import static org.mockito.Mockito.\*;

import static org.junit.Assert.\*;

import org.junit.Test;

public class UserServiceTest {

@Test

public void testGetUserEmailUpperCase() {

UserDao mockDao = mock(UserDao.class);

when(mockDao.getUserEmail(1)).thenReturn("keerthi@example.com");

UserService userService = new UserService(mockDao);

String result = userService.getUserEmailUpperCase(1);

assertEquals("KEERTHI@EXAMPLE.COM", result);

verify(mockDao).getUserEmail(1);

}

}

**OUTPUT:**



**Exercise 2: Verifying Interactions**

**EmailService.java**

**Scenario:** A dependency we will mock. Represents an email-sending component.

package com.example.verify;

public interface EmailService {

void sendWelcomeEmail(String email);

}

**UserRegistrationService.java**

package com.example.verify;

public class UserRegistrationService {

private EmailService emailService;

public UserRegistrationService(EmailService emailService) {

this.emailService = emailService;

}

public void registerUser(String email) {

// Simulate saving user (omitted)

emailService.sendWelcomeEmail(email);

}

}

**UserRegistrationServiceTest.java**

package com.example.verify;

import static org.mockito.Mockito.\*;

import org.junit.Test;

public class UserRegistrationServiceTest {

@Test

public void testSendWelcomeEmailCalled() {

// Create mock of EmailService

EmailService mockEmailService = mock(EmailService.class);

// Inject mock into UserRegistrationService

UserRegistrationService registrationService = new UserRegistrationService(mockEmailService);

// Perform action

registrationService.registerUser("keerthi@example.com");

// Verify that sendWelcomeEmail() was called once with expected argument

verify(mockEmailService, times(1)).sendWelcomeEmail("keerthi@example.com");

}

}

**OUTPUT:**

