**WEEK 2  
PLSQL Programming,** **TDD using JUnit5 and Mockito**

**Superset ID:- 4993066**

1. **JUnit Exercise 1 – Setting Up JUnit:-**

The test checks if 2 + 3 equals 5. It runs fine, so the console shows 1 test passed, with no errors.The goal is to run a simple junit test class

Calculator.Java:-

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

CalculatorTest.java:

import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.Test;

public class CalculatorTest {

@Test

void testAdd() {

Calculator calc = new Calculator();

int result = calc.add(2, 3);

assertEquals(5, result);

}

}

**Output:**

**-------------------------------------------------------**

**T E S T S**

**-------------------------------------------------------**

**Running CalculatorTest**

2)JUnit Exercise 3 – Assertions in JUnit:

This test checks if strings are empty or not. All checks pass, so the output says 1 test ran successfully.

**StringUtils.java -**

public class StringUtils {

public boolean isEmpty(String str) {

return str == null || str.isEmpty();

}

}

**StringUtilsTest.java: -**

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

import org.junit.jupiter.api.Test;

public class StringUtilsTest {

@Test

void testIsEmpty() {

StringUtils util = new StringUtils();

assertTrue(util.isEmpty(null));

assertTrue(util.isEmpty(""));

assertFalse(util.isEmpty("hello"));

}

}  
  
Output:  
-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running StringUtilsTest

**3)JUnit Exercise 4 – AAA Pattern, Fixtures, Setup, Teardown:-**

Before running the test, it prints “Setup complete.” It multiplies numbers correctly, prints “Teardown complete,” and the test passes without errors.

**MathUtil.java:-**

public class MathUtil {

public int multiply(int a, int b) {

return a \* b;

}

}

**MathUtilTest.java:-**

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

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

public class MathUtilTest {

MathUtil util;

@BeforeEach

void setUp() {

util = new MathUtil();

System.out.println("Setup complete.");

}

@AfterEach

void tearDown() {

System.out.println("Teardown complete.");

}

@Test

void testMultiply(){

int a = 3;

int b = 4;

int result = util.multiply(a, b);

assertEquals(12, result);

}

}  
  
Output:

Setup complete.

Teardown complete.

-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running MathUtilTest

**4)** **Mockito Exercise 1 – Mocking and Stubbing :**

A fake user service is created. When asked for an email, it returns a mock email. The test passes because the fake value matches the expected result.

**UserService.java*:-***

public interface UserService {

String getUserEmail(int userId);

}

**UserController.java:-**

public class UserController {

private UserService userService;

public UserController(UserService userService) {

this.userService = userService;

}

public String fetchUserEmail(int userId) {

return userService.getUserEmail(userId);

}

}

**UserControllerTest.java:-**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class UserControllerTest {

@Test

void testFetchUserEmail() {

UserService userServiceMock = mock(UserService.class);

when(userServiceMock.getUserEmail(1)).thenReturn("test@example.com");

UserController controller = new UserController(userServiceMock);

String email = controller.fetchUserEmail(1);

assertEquals("test@example.com", email);

}

}

Output:

-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running UserControllerTest

**5) Mockito Exercise 2 – Verifying Interactions:**

Similar to before, but also checks if the method was really called. The test passes, confirming the interaction happened.

**UserControllerTest\_Verify.java:-**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

public class UserControllerTest\_Verify {

@Test

void testVerifyInteraction() {

UserService userServiceMock = mock(UserService.class);

when(userServiceMock.getUserEmail(2)).thenReturn("mock@domain.com");

UserController controller = new UserController(userServiceMock);

String email = controller.fetchUserEmail(2);

assertEquals("mock@domain.com", email);

verify(userServiceMock).getUserEmail(2);

}

}  
Output:

-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running UserControllerTest\_Verify

**6) SLF4J Exercise 1 – Logging Error and Warning Levels**

The program logs an info, warning, and error message. The console shows all three messages with different log levels.

**LoggingDemo.java:-**

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggingDemo {

private static final Logger logger = LoggerFactory.getLogger(LoggingDemo.class);

public static void main(String[] args) {

logger.info("Hello from SLF4J");

logger.warn("This is a warning!");

logger.error("An error has occurred!");

}

}

Output:

[main] INFO LoggingDemo - Hello from SLF4J

[main] WARN LoggingDemo - This is a warning!

[main] ERROR LoggingDemo - An error has occurred!