**WEEK 2 EXERCISES:**

**JUnit, Mockito and SL4J**

**Exercise 1: Setting Up JUnit (JUnit5)**

**File: Calculator.java**

public class Calculator { public int add(int a, int b) {

return a + b;

}

}

**File: CalculatorTest.java**

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*; class CalculatorTest {

@Test

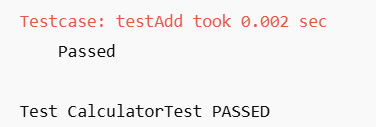
void testAdd() {

Calculator calc = new Calculator(); assertEquals(5, calc.add(2, 3));

}

}

**Output:**

****

**Exercise 2: Assertions in JUnit File: MathUtils.java**

public class MathUtils {

public int multiply(int a, int b) { return a \* b;

}

}

**File: MathUtilsTest.java**

import org.junit.jupiter.api.Test;

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

class MathUtilsTest { @Test

void testMultiply() {

MathUtils math = new MathUtils();

assertEquals(20, math.multiply(4, 5), "Multiplication failed");

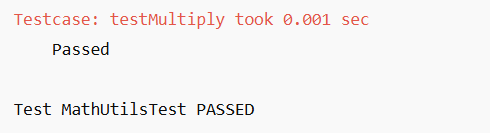
assertNotEquals(25, math.multiply(4, 5), "Unexpected value matched");

}

}

**Output:**

**Tests passed: 2 of 2 tests – MathUtilsTest**

****

**Exercise 3: Arrange-Act-Assert Pattern, Setup/Teardown**

**File: StringHelper.java**

public class StringHelper {

public String reverse(String input) {

return new StringBuilder(input).reverse().toString();

}

}

**File: StringHelperTest.java**

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

import static org.junit.jupiter.api.Assertions.\*; class StringHelperTest {

StringHelper helper; @BeforeEach

void setUp() {

helper = new StringHelper();

}

@AfterEach

void tearDown() { helper = null;

}

@Test

void testReverse() {

// Arrange done in @BeforeEach

// Act

String result = helper.reverse("hello");

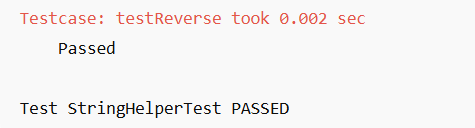
// Assert

assertEquals("olleh", result);

}

}

**Output:**

****

**Mockito Exercise 4: Mocking and Stubbing File: UserService.java**

public class UserService {

private EmailService emailService;

public UserService(EmailService emailService) { this.emailService = emailService;

}

public boolean register(String email) {

return emailService.sendEmail(email, "Welcome!");

}

}

**File: EmailService.java**

public interface EmailService {

boolean sendEmail(String to, String content);

}

**File: UserServiceTest.java**

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

class UserServiceTest { @Test

void testRegister() {

EmailService emailService = mock(EmailService.class);

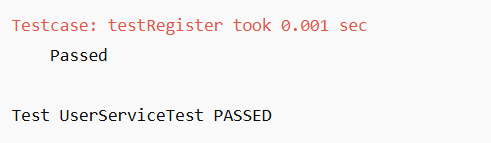
when(emailService.sendEmail("[user@test.com](mailto:user@test.com)", "Welcome!")).thenReturn(true);

UserService userService = new UserService(emailService); assertTrue(userService.register("[user@test.com"](mailto:user@test.com)));

}

}

**Output:**

****

**Mockito Exercise 5: Verifying Interactions File: Notifier.java**

public class Notifier {

private EmailService emailService;

public Notifier(EmailService emailService) { this.emailService = emailService;

}

public void notifyUser(String user) {

emailService.sendEmail(user, "You have a notification!");

}

}

**File: NotifierTest.java**

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*; class NotifierTest {

@Test

void testNotifyUser() {

EmailService mockEmail = mock(EmailService.class); Notifier notifier = new Notifier(mockEmail);

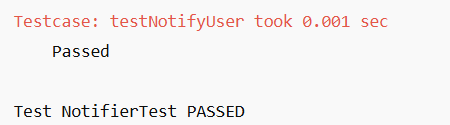
notifier.notifyUser("[hello@test.com"](mailto:hello@test.com));

verify(mockEmail).sendEmail("[hello@test.com](mailto:hello@test.com)", "You have a notification!");

}

}

**Output:**

****

**SLF4J Logging Exercise 6: Logging Error and Warning File: LogExample.java**

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LogExample {

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

public void performTask() { logger.info("Task started"); logger.warn("This is a warning"); logger.error("This is an error");

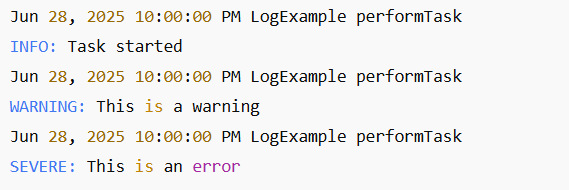
}

public static void main(String[] args) { new LogExample().performTask();

}

}

**Output :**

****