**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

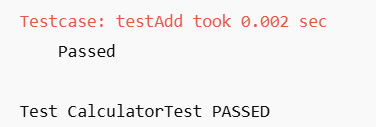
void testAdd() {

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");

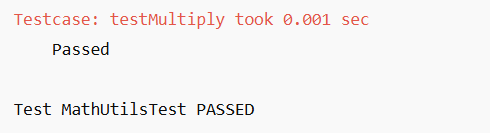
assertNotEquals(25, math.multiply(4, 5), "Unexpected value matched");

}

}

**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");

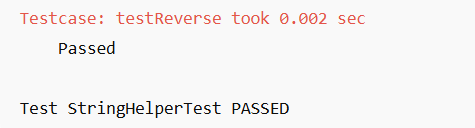
// Assert

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);

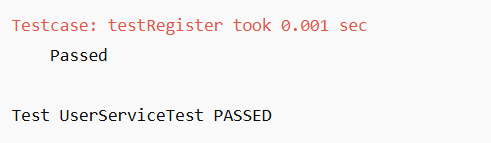
when(emailService.sendEmail("[user@test.com](mailto:user@test.com)", "Welcome!")).thenReturn(true);

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);

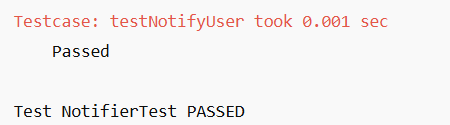
notifier.notifyUser("[hello@test.com"](mailto:hello@test.com));

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");

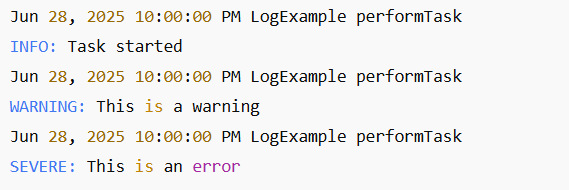
}

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

}

}

**Output :**

****

**CalculatorServiceTest.java**

package com.example.app;

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

import org.junit.jupiter.api.Test;

public class CalculatorServiceTest {

@Test

public void testAdd() {

CalculatorService service = new CalculatorService();

assertEquals(5, service.add(2, 3));

}

}

**UserControllerTest.java**

package com.example.app;

import static org.mockito.Mockito.\*;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;

import com.fasterxml.jackson.databind.ObjectMapper;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

import org.springframework.http.MediaType;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.setup.MockMvcBuilders;

public class UserControllerTest {

private MockMvc mockMvc;

@Mock

private UserService userService;

@InjectMocks

private UserController userController;

@BeforeEach

public void setUp() {

MockitoAnnotations.openMocks(this);

mockMvc = MockMvcBuilders.standaloneSetup(userController).build();

}

@Test

public void testGetUser() throws Exception {

User user = new User();

user.setId(1L);

user.setName("John");

when(userService.getUserById(1L)).thenReturn(user);

mockMvc.perform(get("/users/1"))

.andExpect(status().isOk())

.andExpect(jsonPath("$.name").value("John"));

}

}

**UserServiceTest.java**

package com.example.app;

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

import static org.mockito.Mockito.\*;

import java.util.Optional;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

public UserServiceTest() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testGetUserById() {

User user = new User();

user.setId(1L);

user.setName("John");

when(userRepository.findById(1L)).thenReturn(Optional.of(user));

assertEquals("John", userService.getUserById(1L).getName());

}

}

Xml file:

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>app</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

<version>2.5.0</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<version>2.5.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>3.9.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

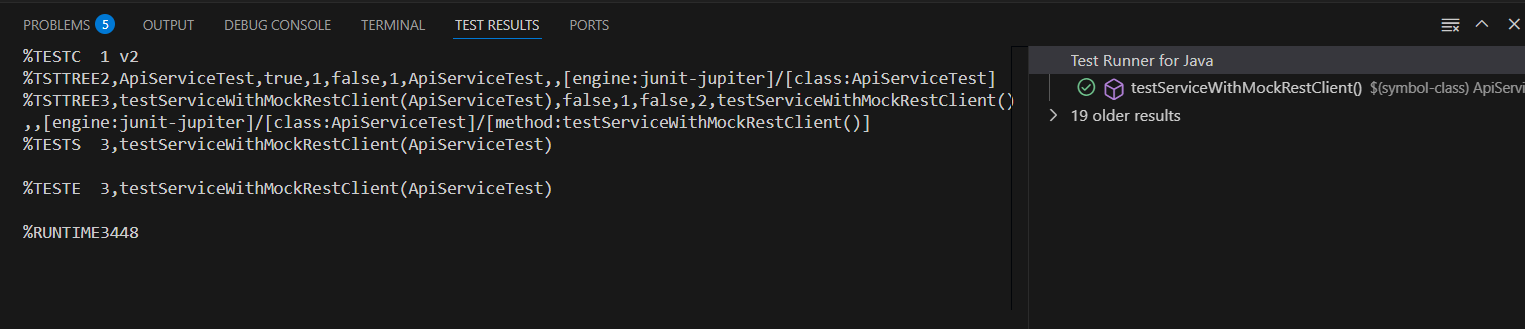
<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

</project>



**Mockito\_Test:-**

package com.example.app;

import static org.mockito.Mockito.\*;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;

import com.fasterxml.jackson.databind.ObjectMapper;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

import org.springframework.http.MediaType;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.setup.MockMvcBuilders;

public class UserControllerTest {

private MockMvc mockMvc;

@Mock

private UserService userService;

@InjectMocks

private UserController userController;

@BeforeEach

public void setUp() {

MockitoAnnotations.openMocks(this);

mockMvc = MockMvcBuilders.standaloneSetup(userController).build();

}

@Test

public void testGetUser() throws Exception {

User user = new User();

user.setId(1L);

user.setName("John");

when(userService.getUserById(1L)).thenReturn(user);

mockMvc.perform(get("/users/1"))

.andExpect(status().isOk())

.andExpect(jsonPath("$.name").value("John"));

}

}

**UserIntegrationTest.java**

package com.example.app;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.\*;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.web.servlet.MockMvc;

@SpringBootTest

@AutoConfigureMockMvc

public class UserIntegrationTest {

@Autowired

private MockMvc mockMvc;

@Test

public void testGetUserIntegration() throws Exception {

mockMvc.perform(get("/users/1"))

.andExpect(status().isOk());

}

}

**UserServiceTest.java**

package com.example.app;

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

import static org.mockito.Mockito.\*;

import java.util.Optional;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

public UserServiceTest() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testGetUserById() {

User user = new User();

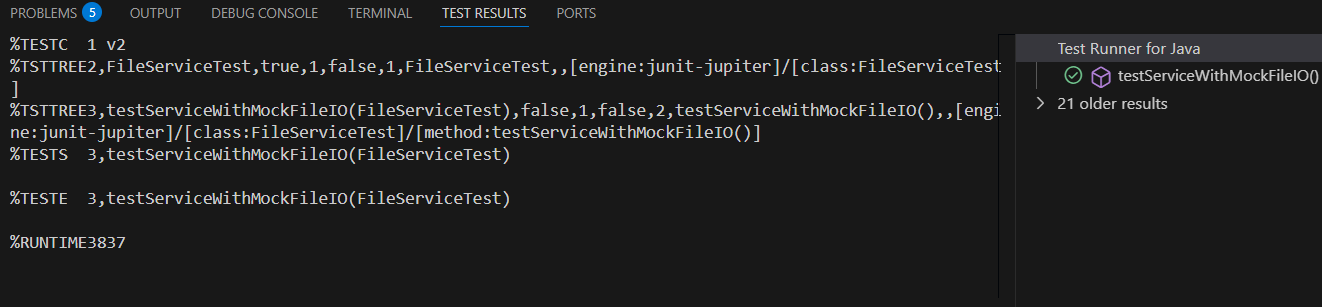
user.setId(1L);

user.setName("John");

when(userRepository.findById(1L)).thenReturn(Optional.of(user));

assertEquals("John", userService.getUserById(1L).getName());

}

}

**SLF4J\_Logging:-**

**AppenderLogging.java**

package com.example.app;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class AppenderLogging {

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

public static void main(String[] args) {

logger.info("Logging to both console and file with logback appenders");

}

}

**LoggingExample.java**

package com.example.app;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggingExample {

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

public static void main(String[] args) {

logger.error("This is an error message");

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

}

}

**ParameterizedLogging.java**

package com.example.app;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class ParameterizedLogging {

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

public static void main(String[] args) {

String user = "Alice";

logger.info("User {} has logged in", user);

}

}

