**TDD using JUnit5 and Mockito**

**1. JUnit\_Basic Testing Exercises**

**Exercise 1: Setting Up Junit  
code:**

import org.junit.jupiter.api.Test;

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

class Calculator {

public int add(int a, int b) {

return a + b;

}

}

public class CalculatorTest {

@Test

void testAdd() {

Calculator calc = new Calculator();

int result = calc.add(4, 6);

assertEquals(10, result);

}

}  
  
**Exercise 3: Assertions in Junit  
code:**  
import org.junit.jupiter.api.Test;

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

public class AssertionsTest {

@Test

void testAssertions() {

assertEquals(10, 7 + 3);

assertTrue(9 > 4);

assertFalse(3 > 7);

assertNull(null);

assertNotNull("JUnit5");

}

}  
 **Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in Junit  
code:**

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.AfterEach;

import org.junit.jupiter.api.Test;

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

class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

public class CalculatorLifecycleTest {

private Calculator calculator;

@BeforeEach

void setUp() {

calculator = new Calculator();

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

}

@AfterEach

void tearDown() {

calculator = null;

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

}

@Test

void testAdd() {

assertEquals(15, calculator.add(10, 5));

}

@Test

void testSubtract() {

assertEquals(6, calculator.subtract(10, 4));

}

}

**3. Mockito exercises  
  
Exercise 1: Mocking and Stubbing  
code:**

import org.junit.jupiter.api.Test;

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

import static org.mockito.Mockito.\*;

interface ExternalApi {

String getData();

}

class MyService {

private final ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

public class MyServiceMockingTest {

@Test

void testExternalApiMockingAndStubbing() {

ExternalApi mockApi = mock(ExternalApi.class);

when(mockApi.getData()).thenReturn("Stubbed Data");

MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Stubbed Data", result);

}

}

**Exercise 2: Verifying Interactions  
code:**

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceVerificationTest {

@Test

void testVerifyInteraction() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi, times(1)).getData();

}

}

**6. SL4J Logging exercises**

**Exercise 1: Logging Error Messages and Warning Levels   
code:  
pom.xml:**

<dependencies>

<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>  
  
**Java class code:**

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("Critical error occurred while processing payment");

logger.warn("Low disk space warning for backup storage");

}

}