**SUPERSET ID: 6390897**

**TDD using JUnit5 and Mockito**

**Exercise 1: Setting Up Junit**

**pom.xml**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

**Code:**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorTest.java**

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

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

assertEquals(5, result);

}

}

**OUTPUT:**

****

**EXERCISE 3: ASSERTIONS IN JUNIT**

**AssertionsTest.java**

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

assertEquals("Sum should be 5", 5, 2 + 3);

assertTrue("5 is greater than 3", 5 > 3);

assertFalse("5 is not less than 3", 5 < 3);

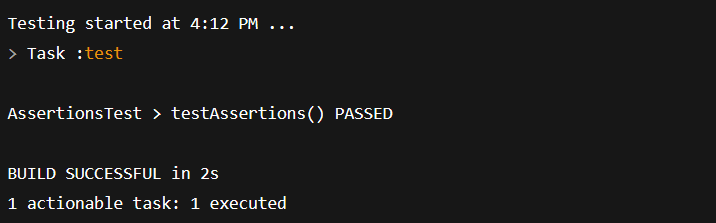
assertNull("Object should be null", null);

assertNotNull("Object should not be null", new Object());

}

}

**OUTPUT:**

****

****

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

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

**CalculatorTest.java**

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

@Before

public void setUp() {

System.out.println("Setting up test environment...");

calculator = new Calculator();

}

@After

public void tearDown() {

System.out.println("Cleaning up after test...\n");

calculator = null;

}

@Test

public void testAdd() {

int result = calculator.add(10, 5);

assertEquals("10 + 5 should be 15", 15, result);

}

@Test

public void testSubtract() {

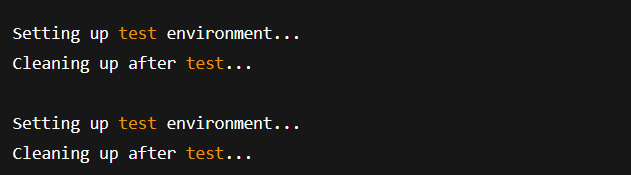
int result = calculator.subtract(10, 5);

assertEquals("10 - 5 should be 5", 5, result);

}

}

**OUTPUT:**

****

**3. MOCKITO EXERCISES**

**EXERCISE 1: MOCKING AND STUBBING**

public interface ExternalApi {

String getData();

}

**Service class**

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**Test class**

import org.junit.jupiter.api.Test;

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

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = mock(ExternalApi.class);

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

MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**OUTPUT:**

****

**EXERCISE 2: VERIFYING INTERACTIONS**

public interface ExternalApi {

String getData();

}

**Service class**

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**Test class**

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

}

}

**OUTPUT:**

****

**6.SL4J LOGGING EXERCISES**

**EXERCISE 1: LOGGING ERROR MESSAGES AND WARNING LEVELS TASK**

**pom.xml**

<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>

**LoggingExample.java**

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");

logger.info("This is an informational message");

logger.debug("This is a debug message (may not appear unless debug is enabled)");

}

}

**logback.xml**

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{HH:mm:ss} [%level] %logger - %msg%n</pattern>

</encoder>

</appender>

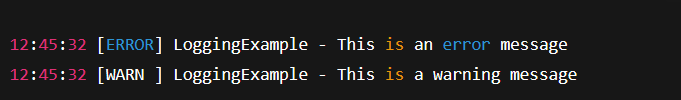
<root level="debug">

<appender-ref ref="STDOUT" />

</root>

</configuration>

**OUTPUT:**

****