Week 2: TDD Using JUnit5 & Mockito

**JUnit5 Basics – Exercise 1: Setting Up JUnit**

# Scenario:

You need to set up JUnit in your Java project to start writing unit tests.

# Procedure:

# Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).

# Add JUnit dependency to your project.

# Create a new test class in your project.

# Implementation:

//EvenChecker.java

package org.example;

public class EvenChecker {

    public boolean isEven(int number) {

        return number % 2 == 0;

    }

}

//EvenCheckerTest.java

package org.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class EvenCheckerTest {

    EvenChecker checker = new EvenChecker();

    @Test

    public void testEven() {

        assertTrue(checker.isEven(4));

    }

    @Test

    public void testOdd() {

        assertFalse(checker.isEven(5));

    }

    @Test

    public void testZero() {

        assertTrue(checker.isEven(0));

    }

}

# Output:

# A screenshot of a computer program AI-generated content may be incorrect.

**JUnit5 Basics – Exercise 3: Assertions in JUnit**

# Scenario:

You need to write basic JUnit tests for a simple Java class.

# Procedure:

# Write tests using various JUnit assertions

public class AssertionsTest {

    @Test

    public void testAssertions() {

        // Assert null

        assertNull(null);

        // Assert not null

        assertNotNull(new Object());

        // Assert true

        assertTrue(5 > 3);

        // Assert false

        assertFalse(5 < 3);

        // Assert equals

        assertEquals(5, 2 + 3);

    }

}

# Implementation:

import org.junit.jupiter.api.Test;

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

public class AssertionsTest {

    @Test

    public void testAssertions() {

        System.out.println("Testing assertEquals");

        assertEquals(5, 2 + 3);

        System.out.println("Testing assertTrue");

        assertTrue(5 > 3);

        System.out.println("Testing assertFalse");

        assertFalse(5 < 3);

        System.out.println("Testing assertNull");

        assertNull(null);

        System.out.println("Testing assertNotNull");

        assertNotNull(new Object());

        System.out.println("All assertions passed.");

    }

}

# Output:

# A screenshot of a computer program AI-generated content may be incorrect.

**JUnit5 Basics – Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

# Scenario:

You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

# Procedure:

* Write tests using the AAA pattern.
* Use @Before and @After annotations for setup and teardown methods.

# Implementation:

//Source Class

public class EvenChecker {

    public boolean isEven(int number) {

        return number % 2 == 0;

    }

}

//Test Class

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class EvenNumCheckerTest {

    private EvenChecker evenChecker;

    @Before

    public void setUp() {

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

        evenChecker = new EvenChecker();

    }

    @After

    public void tearDown() {

        System.out.println("Tearing down...");

        evenChecker = null;

    }

    @Test

    public void testEvenNumber() {

        boolean result = evenChecker.isEven(10);

        assertTrue(result);

    }

    @Test

    public void testOddNumber() {

        boolean result = evenChecker.isEven(5);

        assertFalse(result);

    }

}

# Output:

# A screenshot of a computer program AI-generated content may be incorrect.

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

# Scenario:

You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.

# Procedure:

* Create a mock object for the external API.
* Stub the methods to return predefined values.
* Write a test case that uses the mock object.

**Implementation:**

//ExternalAPI

public interface ExternalApi {

    String getData();

}

//MyService Class

public class MyService {

    private ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

//MyServiceTest Class

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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

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();

        System.out.println("Fetched data from mock: " + result);

        assertEquals("Mock Data", result);

    }

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Mockito – Exercise 2: Verifying Interactions**

# Scenario:

You need to ensure that a method is called with specific arguments.

# Procedure:

* Create a mock object.
* Call the method with specific arguments.
* Verify the interaction.

**Implementation:**

//ExternalAPI

public interface ExternalApi {

    String getData();

}

//MyService Class

public class MyService {

    private ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

//MyServiceTest Class

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

    @Test

    public void testVerifyInteraction() {

        ExternalApi mockApi = mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.fetchData();

        System.out.println("Calling fetchData(), expecting getData() to be invoked on mock");

        verify(mockApi).getData();

        System.out.println("Verified: mockApi.getData() was called!");

    }

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.