JUnit Testing Exercises

# Exercise 1: Setting Up JUnit

Scenario:

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

1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).
2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

1. Create a new test class in your project.

A screenshot of a computer

AI-generated content may be incorrect.

# Exercise 2: Writing Basic JUnit Tests

Scenario:

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

1. Create a new Java class with some methods to test.

**Answer:**

**Operation.java:**

package Junit;

public class Operation {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}

}

1. Write JUnit tests for these methods.

**OperationTest.java:**

package Junit;

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

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

public class OperationTest {

Operation op;

@BeforeEach

public void setUp() {

op = new Operation();

}

@Test

public void testAdd() {

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

*assertEquals*(5, result);

}

@Test

public void testMultiply() {

*assertEquals*(6, op.multiply(2, 3));

}

@AfterEach

public void tearDown() {

op = null;

}

}

# Exercise 3: Assertions in JUnit

Scenario:

You need to use different assertions in JUnit to validate your test results. Steps:

1. Write tests using various JUnit assertions.

**OperationTest.java:**

package Junit;

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

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

public class OperationTest {

Operation op;

@BeforeEach

public void setUp() {

op = new Operation();

}

@Test

public void testAdd() {

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

*assertEquals*(5, result);

}

@Test

public void testMultiply() {

*assertEquals*(6, op.multiply(2, 3));

}

@Test

public void testAssertionsDemo() {

*assertEquals*(10, 5 + 5);

*assertTrue*(4 < 5);

*assertFalse*(3 > 10);

*assertNotNull*(new Object());

*assertNull*(null);

}

@AfterEach

public void tearDown() {

op = null;

}

}

A screenshot of a computer

AI-generated content may be incorrect.

# 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.

Steps:

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

**OperationTest.java:**

package Junit;

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

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

public class OperationTest {

Operation op;

@BeforeEach

public void setUp() {

op = new Operation();

}

@Test

public void testAdd() {

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

*assertEquals*(5, result);

}

@Test

public void testMultiply() {

*assertEquals*(6, op.multiply(2, 3));

}

@Test

public void testAssertionsDemo() {

*assertEquals*(10, 5 + 5);

*assertTrue*(4 < 5);

*assertFalse*(3 > 10);

*assertNotNull*(new Object());

*assertNull*(null);

}

@AfterEach

public void tearDown() {

op = null;

}

}

A screenshot of a computer

AI-generated content may be incorrect.