**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.

**Code:**

**Pom. Xml**

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.example.junit</groupId>

<artifactId>JUnitSetupExample</artifactId>

<version>1.0</version>

<name>JUnitSetupExample</name>

<dependencies>

<!-- JUnit Dependency -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

# Calculator.java

package com.example.junit;

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

package com.example.junit;

import org.junit.Test;

import static org.junit.Assert.*assertEquals*;

public class CalculatorTest {

*@Test*

public void testAddition() {

Calculator calc = new Calculator();

int result = calc.add(10, 20);

*assertEquals*(30, result);

}

*@Test*

public void testSubtraction() {

Calculator calc = new Calculator();

int result = calc.subtract(50, 20);

*assertEquals*(30, result);

}

}

# Output:

# 

# 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.
2. Write JUnit tests for these methods.

**Code:**

**Pom.xml**

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.example.junit</groupId>

<artifactId>JUnitBasicTests</artifactId>

<version>1.0</version>

<name>JUnitBasicTests</name>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**MathOperations.java**

package com.example.junit;

public class MathOperations {

public int multiply(int a, int b) {

return a \* b;

}

public boolean isEven(int number) {

return number % 2 == 0;

}

public int divide(int a, int b) {

if (b == 0) throw new IllegalArgumentException("Cannot divide by zero.");

return a / b;

}

}

**MathOperationsTest.java**

package com.example.junit;

import org.junit.Test;

import static org.junit.Assert.\*;

public class MathOperationsTest {

MathOperations math = new MathOperations();

*@Test*

public void testMultiply() {

*assertEquals*(20, math.multiply(4, 5));

*assertEquals*(0, math.multiply(0, 100));

}

*@Test*

public void testIsEven() {

*assertTrue*(math.isEven(4));

*assertFalse*(math.isEven(7));

}

*@Test*(expected = IllegalArgumentException.class)

public void testDivideByZero() {

math.divide(10, 0); // Should throw exception

}

*@Test*

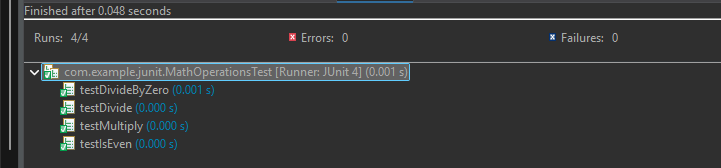
public void testDivide() {

*assertEquals*(5, math.divide(10, 2));

}

}

**Output:**

****

# 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.

Solution Code:

public class AssertionsTest { @Test

public void testAssertions() {

// Assert equals assertEquals(5, 2 + 3);

// Assert true assertTrue(5 > 3);

// Assert false assertFalse(5 < 3);

// Assert null assertNull(null);

// Assert not null assertNotNull(new Object());

}

}

**Code:**

**Pom.xml**

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>JUnitAssertions</artifactId>

<version>1.0</version>

<name>JUnitAssertions</name>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**AssertionsTest.java**

package com.example.junit;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

*@Test*

public void testAssertions() {

// Assert equals

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

// Assert true

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

// Assert false

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

// Assert null

Object obj = null;

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

// Assert not null

Object notNullObj = new Object();

*assertNotNull*("Object should not be null", notNullObj);

// Assert same

String str1 = "hello";

String str2 = str1;

*assertSame*("Should reference same object", str1, str2);

// Assert not same

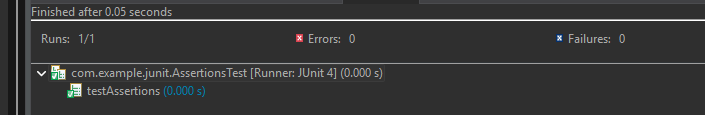
String str3 = new String("hello");

*assertNotSame*("Should not reference same object", str1, str3);

}

}

**Output:**

****

# 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.

**Code:**

**Pom.xml**

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>JUnitExample</artifactId>

<version>1.0</version>

<name>JUnitExample</name>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**AccountService.java**

package com.example.junit;

public class AccountService {

private int balance;

public AccountService(int initialBalance) {

this.balance = initialBalance;

}

public void deposit(int amount) {

if (amount > 0)

balance += amount;

}

public boolean withdraw(int amount) {

if (amount <= balance) {

balance -= amount;

return true;

} else {

return false;

}

}

public int getBalance() {

return balance;

}

}

**AccountServiceTest.java**

package com.example.junit;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AccountServiceTest {

private AccountService account;

// Setup method – runs before each test

*@Before*

public void setUp() {

account = new AccountService(1000); // Arrange

System.***out***.println("Setup complete");

}

// Teardown method – runs after each test

*@After*

public void tearDown() {

System.***out***.println("Teardown complete");

}

*@Test*

public void testDeposit() {

// Act

account.deposit(500);

// Assert

*assertEquals*(1500, account.getBalance());

}

*@Test*

public void testWithdrawSuccess() {

// Act

boolean result = account.withdraw(300);

// Assert

*assertTrue*(result);

*assertEquals*(700, account.getBalance());

}

*@Test*

public void testWithdrawFailure() {

// Act

boolean result = account.withdraw(1500);

// Assert

*assertFalse*(result);

*assertEquals*(1000, account.getBalance());

}

}

**Output:**

