Day 27

Assignment 1: Write a set of JUnit tests for a given class with simple mathematical operations (add, subtract, multiply, divide) using the basic @Test annotation.

A) ***Sure! Below is a set of JUnit tests for a simple Calculator class with methods for adding, subtracting, multiplying, and dividing. These tests use the basic @Test annotation from JUnit 4.***

First, let's assume the Calculator class looks like this:

Java code:

```
public class Calculator {

    public int add(int a, int b) {

        return a + b;

    }

    public int subtract(int a, int b) {

        return a - b;

    }

    public int multiply(int a, int b) {

        return a * b;
```

```
    }

  public int divide(int a, int b) {

        if (b == 0) {

            throw new IllegalArgumentException("Division by zero is
not allowed");

        }

        return a / b;

    }

}
```

Now, here are the JUnit tests for this class using JUnit 4:

Java code:

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

import org.junit.Test;



public class CalculatorTest {

    private final Calculator calculator = new Calculator();

    @Test
```

```
public void testAdd() {
    assertEquals(5, calculator.add(2, 3));
    assertEquals(-1, calculator.add(-2, 1));
    assertEquals(0, calculator.add(0, 0));
}
@Test
public void testSubtract() {
    assertEquals(1, calculator.subtract(3, 2));
    assertEquals(-3, calculator.subtract(-2, 1));
    assertEquals(0, calculator.subtract(0, 0));
}

@Test
public void testMultiply() {
    assertEquals(6, calculator.multiply(2, 3));
    assertEquals(-2, calculator.multiply(-1, 2));
```

```
    assertEquals(0, calculator.multiply(0, 100));
}

@Test
public void testDivide() {
    assertEquals(2, calculator.divide(6, 3));
    assertEquals(-2, calculator.divide(-6, 3));
    assertEquals(1, calculator.divide(7, 7));
}

@Test
public void testDivideByZero() {
    try {
        calculator.divide(1, 0);
        fail("Expected IllegalArgumentException to be thrown");
    } catch (IllegalArgumentException e) {
```

```
            assertEquals("Division by zero is not allowed",
e.getMessage());

        }

    }

}
```

Explanation:

*1. *Setup*:*

- An instance of Calculator is created for use in each test.

*2. *Test Methods*:*

*- *testAdd*: Tests the add method with positive, negative, and zero values.*

- testSubtract: Tests the subtract method with positive, negative, and zero values.

- testMultiply: Tests the multiply method with positive, negative, and zero values.

- testDivide: Tests the divide method with positive and negative values and ensures correct division.

- testDivideByZero: Tests the divide method to ensure it throws an IllegalArgumentException when dividing by zero.

Note:

These tests use the JUnit 4 framework. If you are using JUnit 5, the import statements and test annotations would be slightly different (e.g., @Test would come from org.junit.jupiter.api.Test and assertThrows would be used for exception testing). explanation

Sure! Below is a set of JUnit tests for a simple Calculator class with methods for adding, subtracting, multiplying, and dividing. These tests use the basic @Test annotation from JUnit 4.

First, let's assume the Calculator class looks like this:

java code

```
public class Calculator {

    public int add(int a, int b) {

        return a + b;

    }



    public int subtract(int a, int b) {

        return a - b;

    }



    public int multiply(int a, int b) {
```

```
        return a * b;
    }

    public int divide(int a, int b) {
        if (b == 0) {
            throw new IllegalArgumentException("Division by zero is not allowed");
        }
        return a / b;
    }
}
```

Now, here are the JUnit tests for this class using JUnit 4:

```
java
Copy code
import static org.junit.Assert.*;
import org.junit.Test;
```

```
public class CalculatorTest {

    private final Calculator calculator = new Calculator();

    @Test
    public void testAdd() {
        assertEquals(5, calculator.add(2, 3));
        assertEquals(-1, calculator.add(-2, 1));
        assertEquals(0, calculator.add(0, 0));
    }

    @Test
    public void testSubtract() {
        assertEquals(1, calculator.subtract(3, 2));
        assertEquals(-3, calculator.subtract(-2, 1));
```

```
    assertEquals(0, calculator.subtract(0, 0));
}

@Test
public void testMultiply() {
    assertEquals(6, calculator.multiply(2, 3));
    assertEquals(-2, calculator.multiply(-1, 2));
    assertEquals(0, calculator.multiply(0, 100));
}

@Test
public void testDivide() {
    assertEquals(2, calculator.divide(6, 3));
    assertEquals(-2, calculator.divide(-6, 3));
    assertEquals(1, calculator.divide(7, 7));
}
```

```
    @Test
    public void testDivideByZero() {
        try {
            calculator.divide(1, 0);
            fail("Expected IllegalArgumentException to be thrown");
        } catch (IllegalArgumentException e) {
            assertEquals("Division by zero is not allowed",
e.getMessage());
        }
    }
}
```

Explanation:

Setup:

An instance of Calculator is created for use in each test.

Test Methods:

testAdd: Tests the add method with positive, negative, and zero values.

testSubtract: Tests the subtract method with positive, negative, and zero values.

testMultiply: Tests the multiply method with positive, negative, and zero values.

testDivide: Tests the divide method with positive and negative values and ensures correct division.

testDivideByZero: Tests the divide method to ensure it throws an IllegalArgumentException when dividing by zero.