**Week-2 JUnit Basic Testing Exercises**

1. **Setting Up JUnit**

**Scenario:**

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

**Code:**

**CalculateMultiplication.java**

public class CalculateMultiplication

{

public int multiplication(int n, int m)

{

return n\*m;

}

}

**CalculateMultiplyTest.java**

import org.junit.jupiter.api.Test;

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

public class CalculateMultiplyTest

{

@Test

void testMultiply()

{

CalculateMultiplication c = new CalculateMultiplication();

int result = c.multiplication(4, 5);

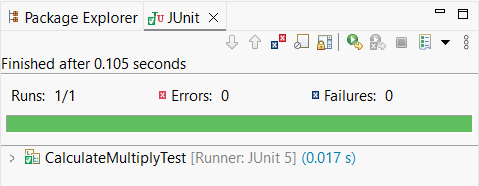
assertEquals(20, result);

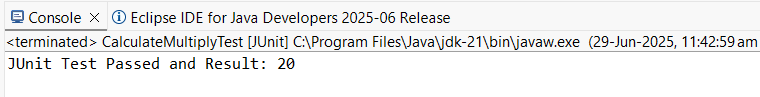
System.out.print("JUnit Test Passed and Result: "+result);

}

}

**Output:**





1. **Assertions in JUnit**

**Scenario:**

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

**Code:**

**AssertionsTest.java**

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

import org.junit.jupiter.api.Test;

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());

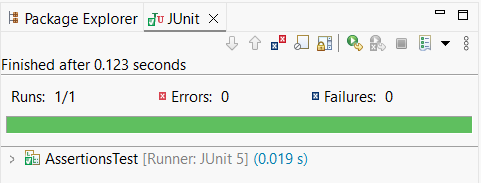
System.out.println("All Assertions passed");

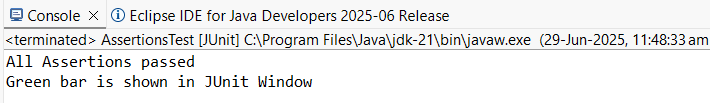
System.out.println("Green bar is shown in JUnit Window");

}

}

**Output:**

****

****

1. **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.

**Code:**

**ArithmeticOperations.java**

public class ArithmeticOperations

{

public int add(int a, int b)

{

return a+b;

}

public int divide(int a, int b)

{

return a/b; // Note: This will throw ArithmeticException if b=0

}

}

**ArithmeticOperationsTest.java**

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

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

public class ArithmeticOperationsTest

{

ArithmeticOperations a;

@BeforeEach // Setup method

public void setup()

{

a = new ArithmeticOperations(); // Arrange

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

}

@AfterEach // Teardown method

public void teardown()

{

a = null; // Shown for learning

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

}

@Test

public void testAddition()

{

// Act

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

// Assert

*assertEquals*(30, result);

}

@Test

public void testDivision()

{

// Act

int result = a.divide(20, 5);

// Assert

*assertEquals*(4, result);

}

@Test

public void testDivideByZero()

{

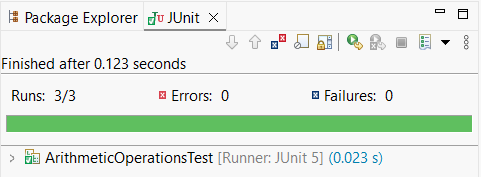
// Act + Assert (expecting exception)

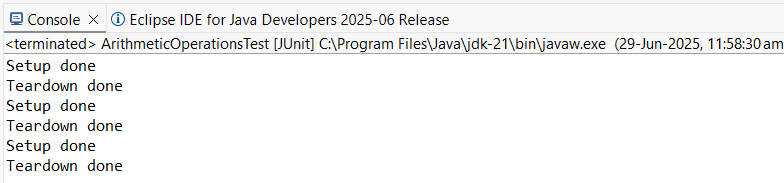
*assertThrows*(ArithmeticException.class, () -> a.divide(10, 0));

}

}

**Output:**

****

****