JUnit Basic Testing

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>
```

3. Create a new test class in your project

CODE:

Calculator.java

```
package com.library.gui;


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 ArithmeticException("Cannot divide by zero");
        }
        return a / b;
    }
}
```

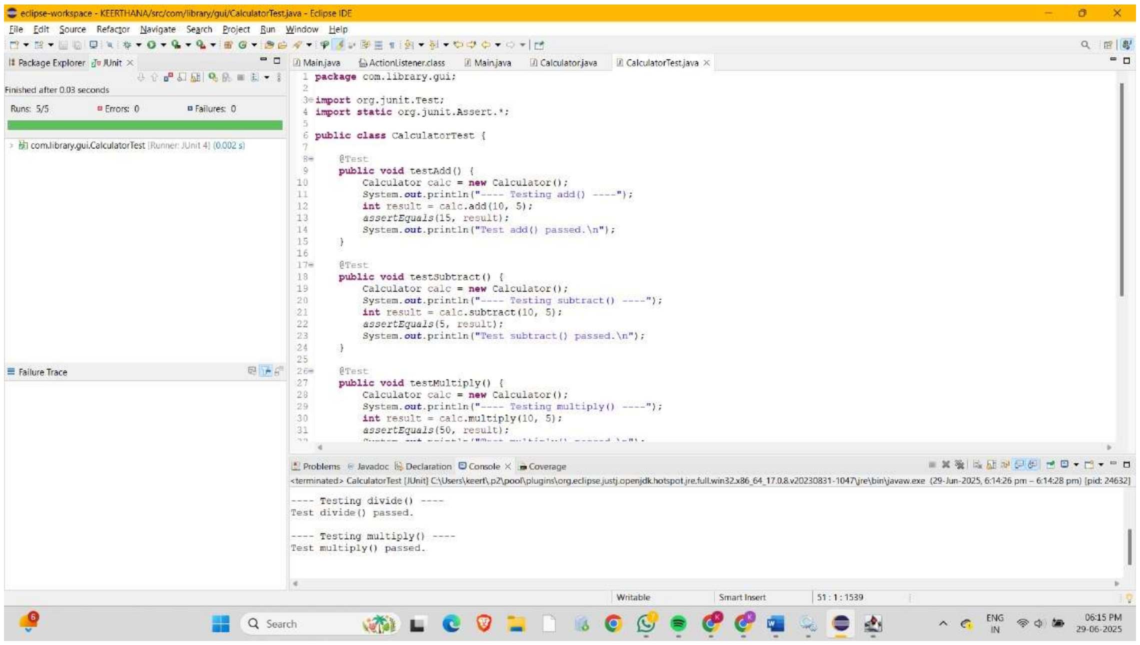
CalculatorTest.java

```
package com.library.gui;

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

public class CalculatorTest {

    @Test
```

```
public void testAdd() {
    Calculator calc = new Calculator();
    System.out.println("---- Testing add() ----");
    int result = calc.add(10, 5);
    assertEquals(15, result);
    System.out.println("Test add() passed.\n");
}

@Test
public void testSubtract() {
    Calculator calc = new Calculator();
    System.out.println("---- Testing subtract() ----");
    int result = calc.subtract(10, 5);
    assertEquals(5, result);
    System.out.println("Test subtract() passed.\n");
}

@Test
public void testMultiply() {
    Calculator calc = new Calculator();
    System.out.println("---- Testing multiply() ----");
    int result = calc.multiply(10, 5);
    assertEquals(50, result);
    System.out.println("Test multiply() passed.\n");
}
```

```
    @Test
    public void testDivide() {
        Calculator calc = new Calculator();
        System.out.println("---- Testing divide() ----");
        int result = calc.divide(10, 5);
        assertEquals(2, result);
        System.out.println("Test divide() passed.\n");
    }

    @Test(expected = ArithmeticException.class)
    public void testDivideByZero() {
        Calculator calc = new Calculator();
        System.out.println("---- Testing divide by zero ----");
        calc.divide(10, 0);  // Expected exception
    }
}
```

OUTPUT:



Finished after 0.03 seconds

Runs: 5/5   Errors: 0   Failures: 0

com.library.gui.CalculatorTest [Runner: JUnit 4] (0.002 s)

```java
package com.library.gui;

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

public class CalculatorTest {

    @Test
    public void testAdd() {
        Calculator calc = new Calculator();
        System.out.println("---- Testing add() ----");
        int result = calc.add(10, 5);
        assertEquals(15, result);
        System.out.println("Test add() passed.\n");
    }

    @Test
    public void testSubtract() {
        Calculator calc = new Calculator();
        System.out.println("---- Testing subtract() ----");
        int result = calc.subtract(10, 5);
        assertEquals(5, result);
        System.out.println("Test subtract() passed.\n");
    }

    @Test
    public void testMultiply() {
        Calculator calc = new Calculator();
        System.out.println("---- Testing multiply() ----");
        int result = calc.multiply(10, 5);
        assertEquals(50, result);
```

```
<terminated> CalculatorTest [JUnit] C:\Users\keert\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe  (29-Jun-2025, 6:14:26 pm – 6:14:28 pm) [pid: 24632]

---- Testing divide() ----
Test divide() passed.

---- Testing multiply() ----
Test multiply() passed.
```

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:

```java
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());
}
}
```

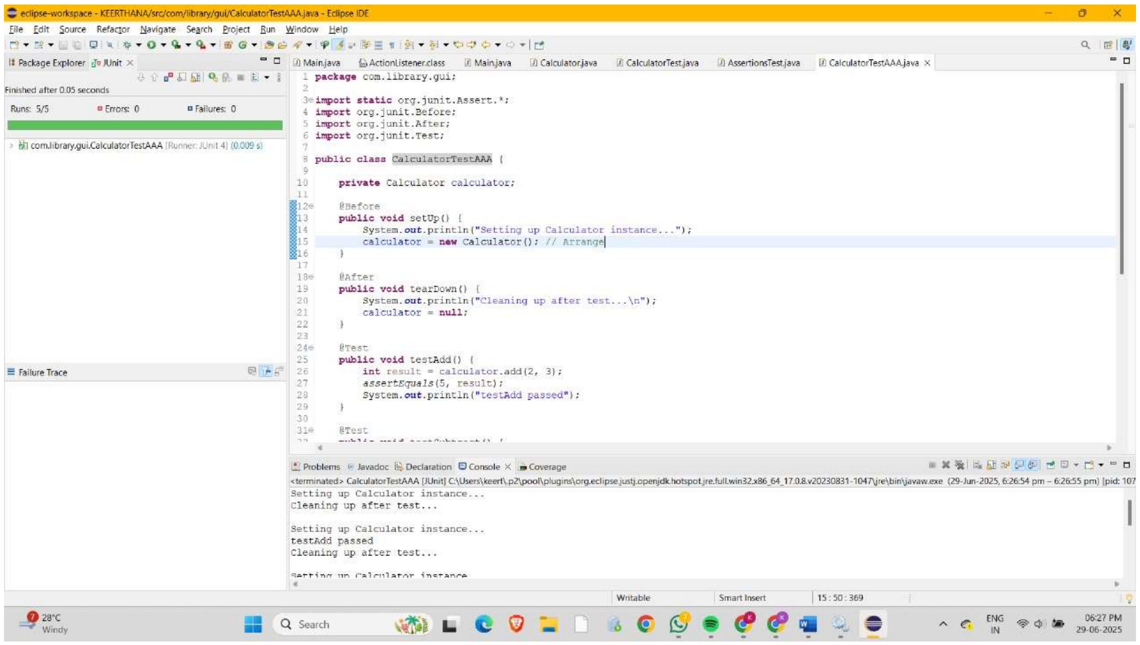
CODE:

AssertionsTest.java

```
package com.library.gui;

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

public class AssertionsTest {

  @Test
  public void testAssertions() {
    System.out.println("Running testAssertions...");

    // Assert equals
    assertEquals("Sum should be 5", 5, 2 + 3);
    System.out.println("assertEquals passed");

    // Assert true
    assertTrue("5 is greater than 3", 5 > 3);
```

```
        System.out.println("assertTrue passed");

        // Assert false
        assertFalse("5 is not less than 3", 5 < 3);
        System.out.println("assertFalse passed");

        // Assert null
        assertNull("Value should be null", null);
        System.out.println("assertNull passed");

        // Assert not null
        assertNotNull("Object should not be null", new Object());
        System.out.println("assertNotNull passed");
    }
}
```

OUTPUT:



Finished after 0.05 seconds

Runs: 5/5 Errors: 0 Failures: 0

com.library.gui.CalculatorTestAAA [Runner: JUnit 4] (0.009 s)

```java
package com.library.gui;

import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.After;
import org.junit.Test;

public class CalculatorTestAAA {

    private Calculator calculator;

    @Before
    public void setUp() {
        System.out.println("Setting up Calculator instance...");
        calculator = new Calculator(); // Arrange
    }

    @After
    public void tearDown() {
        System.out.println("Cleaning up after test...\n");
        calculator = null;
    }

    @Test
    public void testAdd() {
        int result = calculator.add(2, 3);
        assertEquals(5, result);
        System.out.println("testAdd passed");
    }

    @Test
```

```
<terminated> CalculatorTestAAA [JUnit] C:\Users\keert\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe (29-Jun-2025, 6:26:54 pm – 6:26:55 pm) [pid: 107
Setting up Calculator instance...
Cleaning up after test...

Setting up Calculator instance...
testAdd passed
Cleaning up after test...

Setting up Calculator instance
```

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:

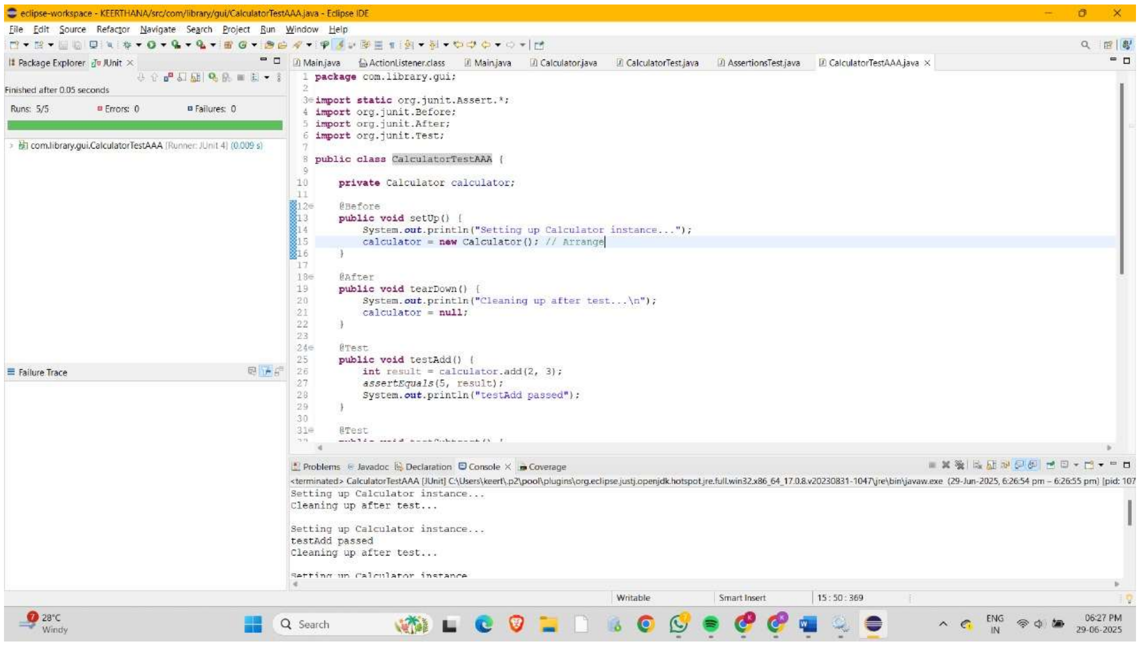
CalculatorTestAAA.java

package com.library.gui;

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

import org.junit.Before;

import org.junit.After;

import org.junit.Test;


public class CalculatorTestAAA {


    private Calculator calculator;


    @Before

    public void setUp() {

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

        calculator = new Calculator(); // Arrange

    }


    @After

    public void tearDown() {

        System.out.println("Cleaning up after test...\n");

        calculator = null;

    }


    @Test

    public void testAdd() {

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

        assertEquals(5, result);

        System.out.println("testAdd passed");
```

```
}

@Test
public void testSubtract() {
    int result = calculator.subtract(10, 4);
    assertEquals(6, result);
    System.out.println("testSubtract passed");
}

@Test
public void testMultiply() {
    int result = calculator.multiply(4, 3);
    assertEquals(12, result);
    System.out.println("testMultiply passed");
}

@Test
public void testDivide() {
    int result = calculator.divide(20, 5);
    assertEquals(4, result);
    System.out.println("testDivide passed");
}

@Test(expected = ArithmeticException.class)
public void testDivideByZero() {
    calculator.divide(10, 0);
    }

}
```

OUTPUT:



eclipse-workspace - KEERTHANA/src/com/library/gui/CalculatorTestAAA.java - Eclipse IDE

Finished after 0.05 seconds

Runs: 5/5 Errors: 0 Failures: 0

com.library.gui.CalculatorTestAAA [Runner: JUnit 4] (0.009 s)

```
package com.library.gui;

import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.After;
import org.junit.Test;

public class CalculatorTestAAA {

    private Calculator calculator;

    @Before
    public void setUp() {
        System.out.println("Setting up Calculator instance...");
        calculator = new Calculator(); // Arrange
    }

    @After
    public void tearDown() {
        System.out.println("Cleaning up after test...\n");
        calculator = null;
    }

    @Test
    public void testAdd() {
        int result = calculator.add(2, 3);
        assertEquals(5, result);
        System.out.println("testAdd passed");
    }

    @Test
```

```
<terminated> CalculatorTestAAA [JUnit] C:\Users\keert\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe (29-Jun-2025, 6:26:54 pm – 6:26:55 pm) [pid: 107
Setting up Calculator instance...
Cleaning up after test...

Setting up Calculator instance...
testAdd passed
Cleaning up after test...
```