**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.

Initially, create a **JUnitDemo** maven project

In that, src/main/java create a **StringUtils.java** file as-

Program:

**public** **class** StringUtils {

**public** **boolean** isPalindrome(String input) {

**if** (input == **null**) **return** **false**;

String reversed = **new** StringBuilder(input).reverse().toString();

**return** input.equalsIgnoreCase(reversed);

}

**public** String reverse(String input) {

**if** (input == **null**) **return** **null**;

**return** **new** StringBuilder(input).reverse().toString();

}

}

Then after, in src/test/java create a **StringUtilsTest.java** as-

**import** org.junit.Test;

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

**public** **class** StringUtilsTest {

@Test

**public** **void** testIsPalindrome() {

StringUtils utils = **new** StringUtils();

*assertTrue*(utils.isPalindrome("madam"));

*assertTrue*(utils.isPalindrome("RaceCar"));

*assertFalse*(utils.isPalindrome("hello"));

*assertFalse*(utils.isPalindrome(**null**));

}

@Test

**public** **void** testReverse() {

StringUtils utils = **new** StringUtils();

*assertEquals*("olleh", utils.reverse("hello"));

*assertEquals*("racecar", utils.reverse("racecar"));

*assertNull*(utils.reverse(**null**));

}

}

And mainly in pom.xml-

<project xmlns="https://maven.apache.org/POM/4.0.0" xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com</groupId>

<artifactId>JUnitDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

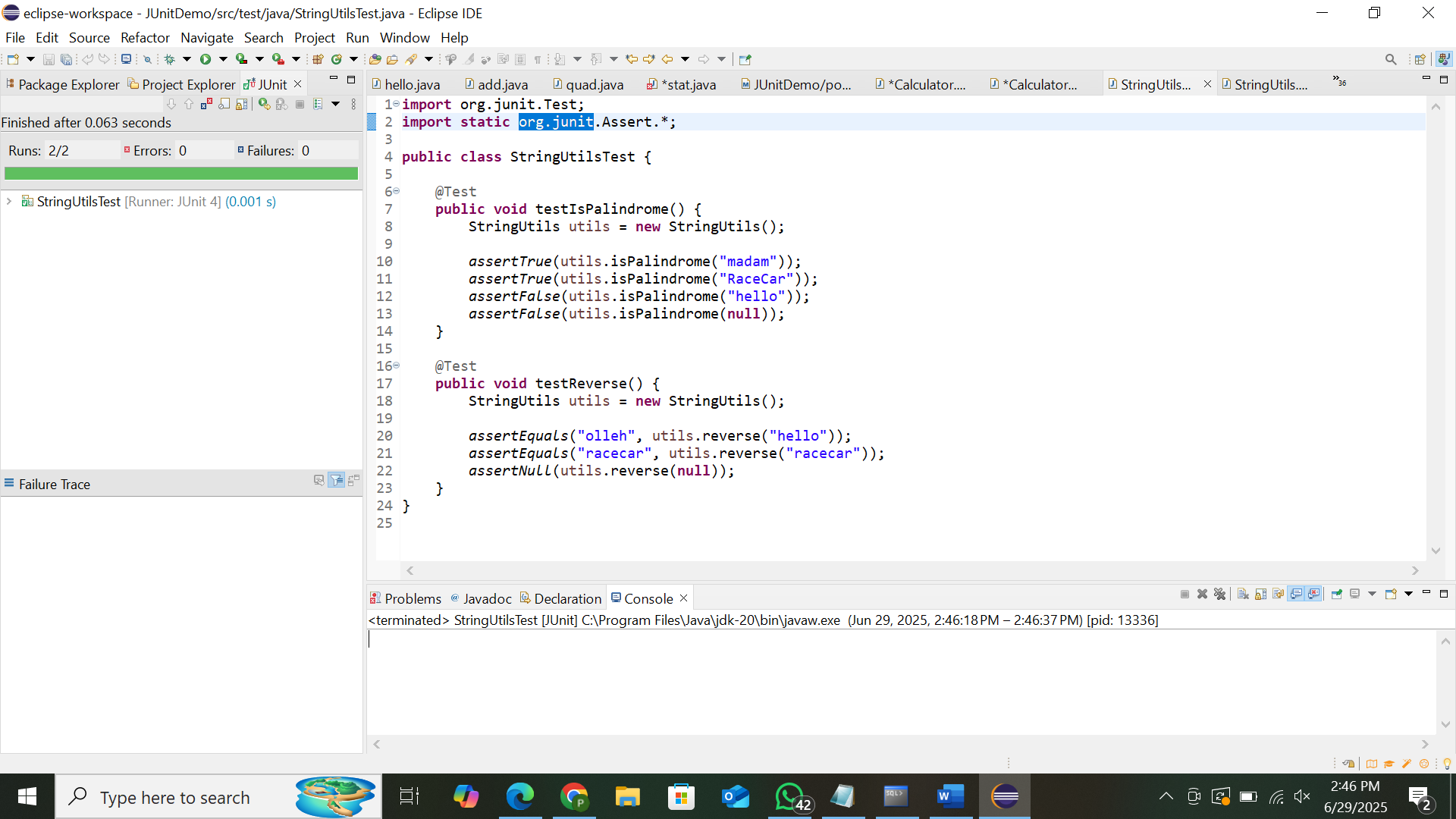
<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency></dependencies></project>

**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.

Create a AssertinsTest file in src/test/java as-

**Program:**

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

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

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

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

Object obj = null;

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

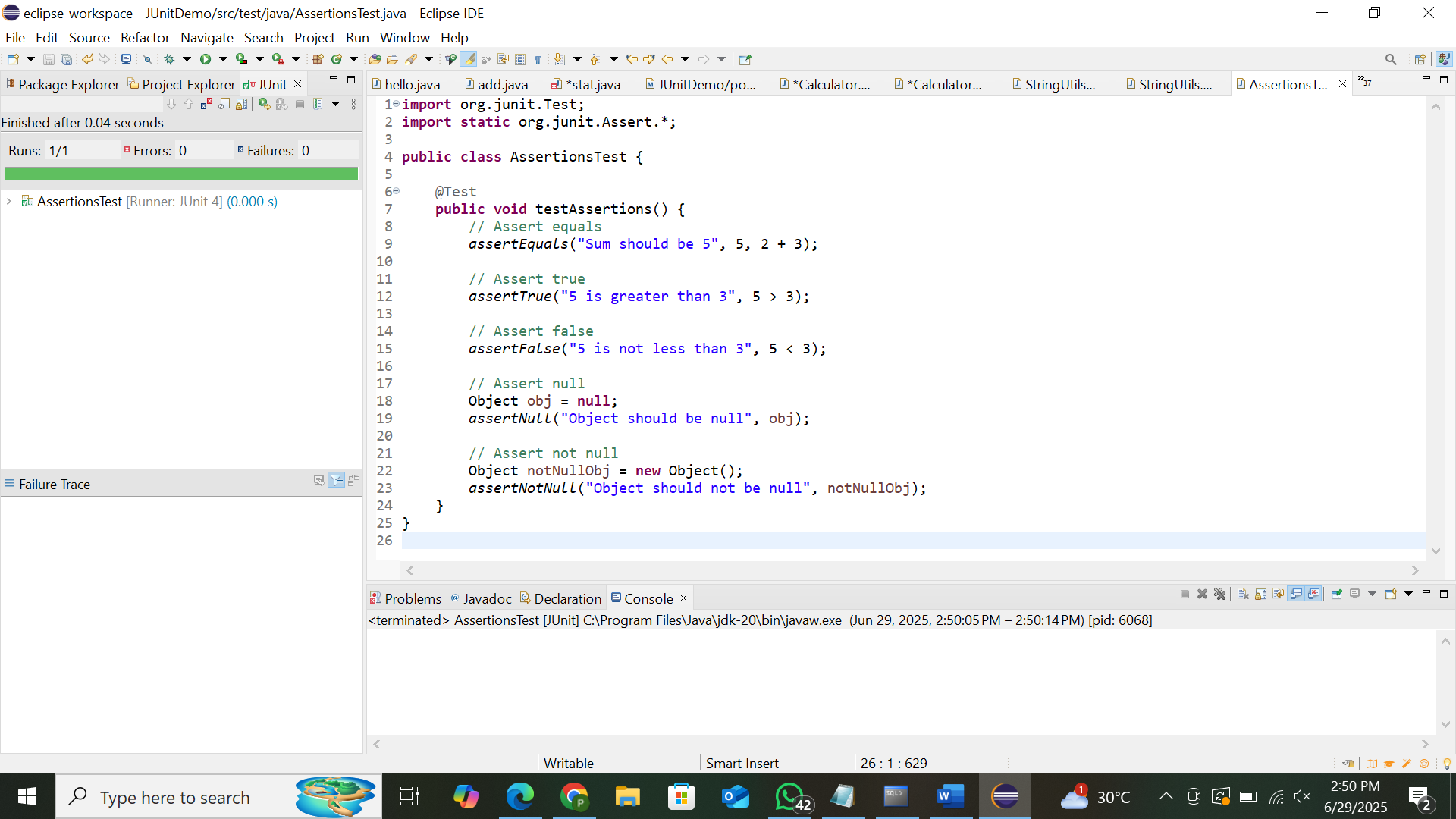
Object notNullObj = new Object();

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

}

}

**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.

Create a CalculatorTestAAA.java in src/test/java as-

**Program:**

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTestAAA {

private Calculator calculator;

@Before

public void setUp() {

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

calculator = new Calculator();

}

@After

public void tearDown() {

System.out.println("Cleaning up...");

calculator = null;

}

@Test

public void testAddition() {

// Arrange

int a = 2;

int b = 3;

int result = calculator.add(a, b);

assertEquals("Addition should be 5", 5, result);

}

@Test

public void testMultiplication() {

// Arrange

int a = 4;

int b = 5;

int result = calculator.multiply(a, b);

assertEquals("Multiplication should be 20", 20, result);

}

}

OUTPUT:

