**Week2\_Junit Basic Testing Exercises**

**SUPERSETID:6393676**

**Exercise 1:**

Setting Up JUnit Scenario: You need to set up JUnit in your Java project to start writing unit tests.

**SOLUTION:**

**Calculator.java**

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**

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

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

assertEquals(5, result);

}

@Test

public void testSubtract() {

Calculator calc = new Calculator();

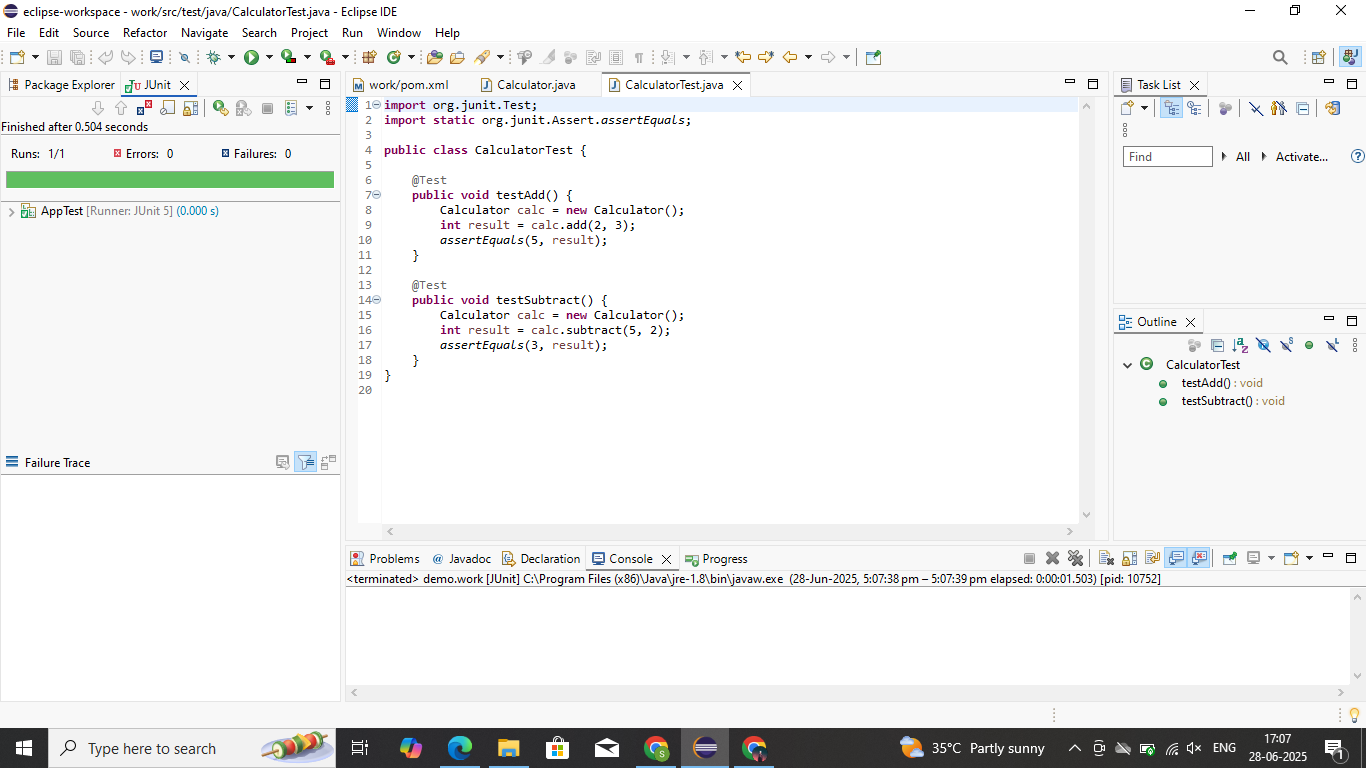
int result = calc.subtract(5, 2);

assertEquals(3, result);

}

}

**OUTPUT:**



**EXERCISE 2:**

Writing Basic JUnit Tests Scenario: You need to write basic JUnit tests for a simple Java class.

**SOLUTION:**

**MathUtils.java**

public class MathUtils {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}

public boolean isEven(int number) {

return number % 2 == 0;

}

}

**MathUtilsTest.java:**

public class MathUtils {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}

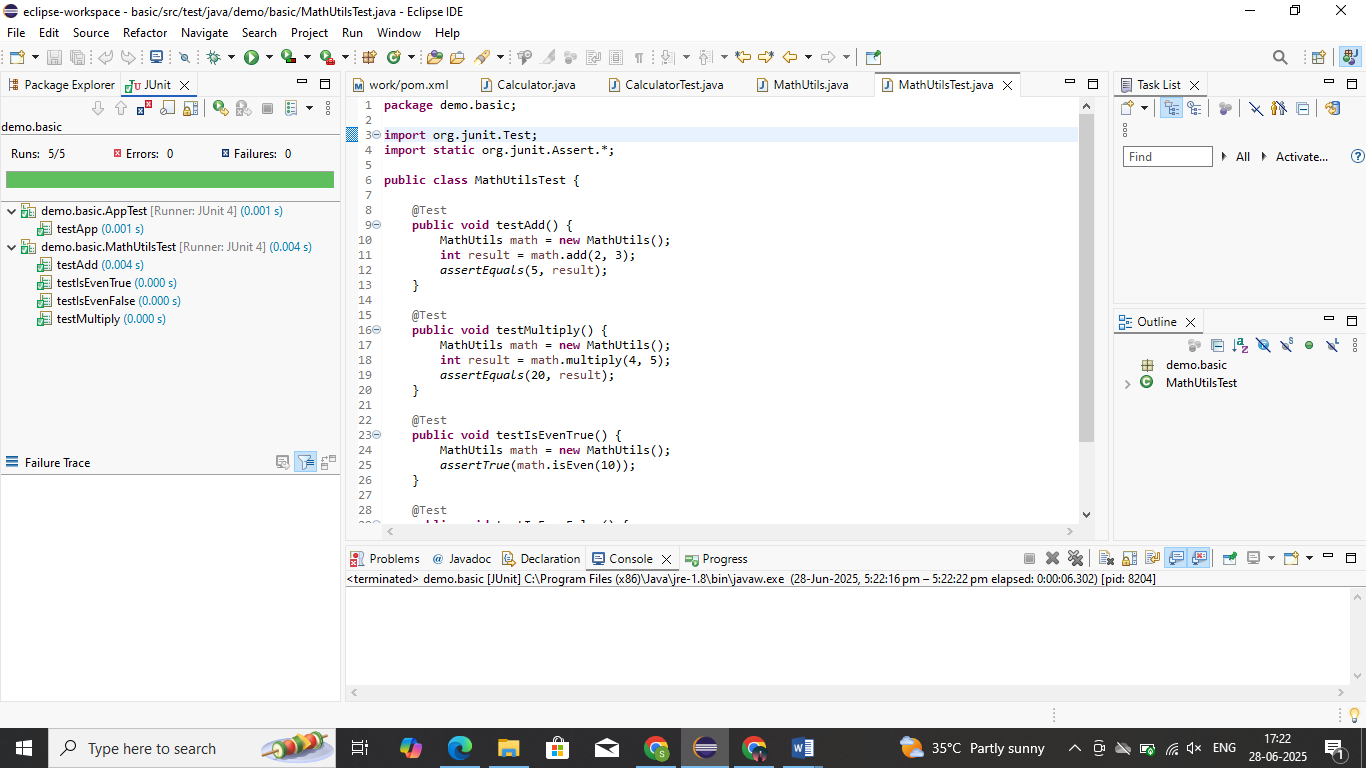
public boolean isEven(int number) {

return number % 2 == 0;

}

}

**OUTPUT:**



**EXERCISE 3:**

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

**SOLUTION:**

**AssertionsTest.java**

import org.junit.Test;

import static org.junit.Assert.\*;

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

Object obj1 = null;

assertNull(obj1);

// Assert not null

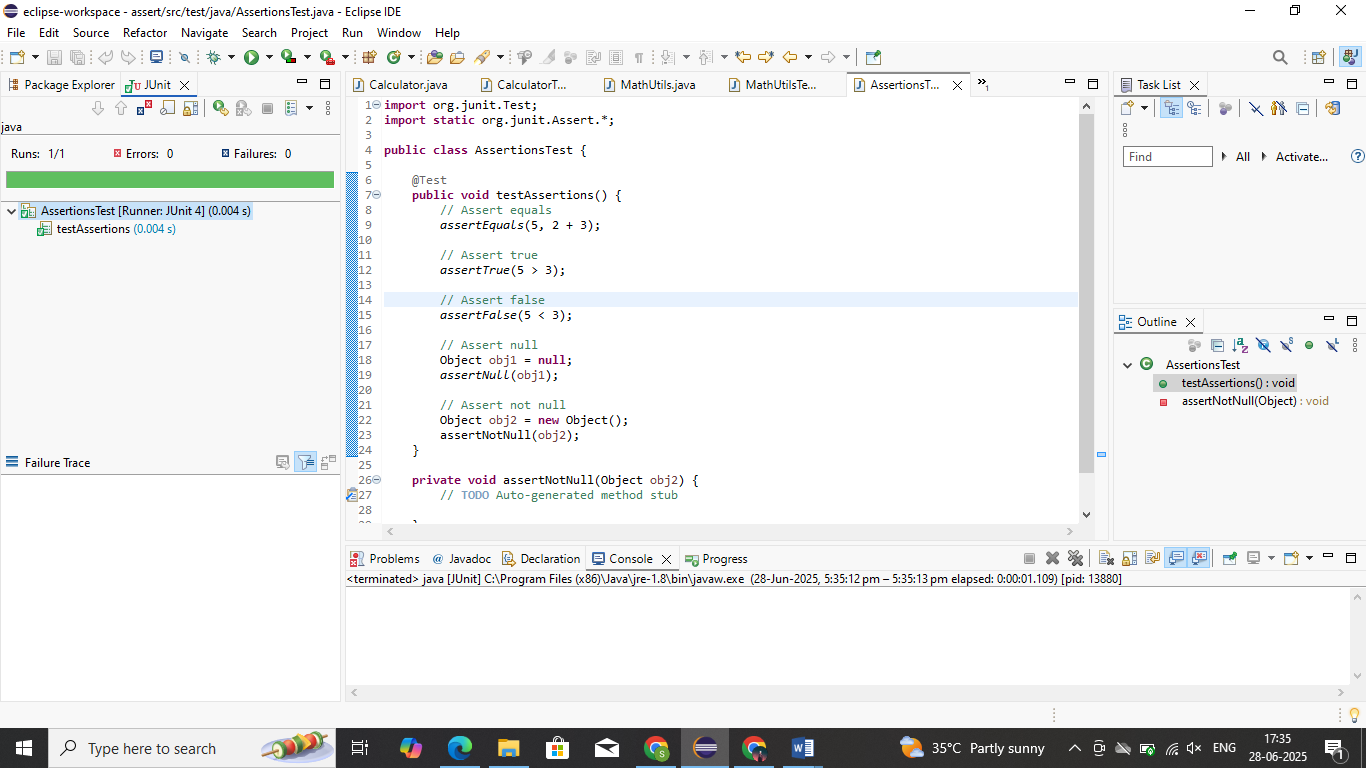
Object obj2 = new Object();

assertNotNull(obj2);

}

}

**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.

**SOLUTION:**

**UserService.java:**

package demo.basic;

public class UserService {

public boolean isValidUsername(String username) {

if (username == null || username.isEmpty()) {

return false;

}

return username.length() >= 5 && username.matches("[a-zA-Z0-9\_]+");

}

}

**UserServiceTest.java:**

package demo.basic;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class UserServiceTest {

private UserService userService;

@Before

public void setUp() {

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

userService = new UserService(); // Arrange

}

@After

public void tearDown() {

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

userService = null;

}

@Test

public void testValidUsername() {

// Act

boolean result = userService.isValidUsername("john\_doe");

// Assert

assertTrue(result);

}

@Test

public void testUsernameTooShort() {

// Act

boolean result = userService.isValidUsername("abc");

// Assert

assertFalse(result);

}

@Test

public void testUsernameWithInvalidCharacters() {

// Act

boolean result = userService.isValidUsername("john@123");

// Assert

assertFalse(result);

}

@Test

public void testNullUsername() {

// Act

boolean result = userService.isValidUsername(null);

// Assert

assertFalse(result);

}

}

**OUTPUT:**

