**WEEK 2 – j-UNIT - 6428652**

**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.**

**Solution:**

package testex1;

public class UnitTest {

public int add(int a,int b) {

return a+b;

}

}

package testex1;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AddNum {

@Test

public void testCorrect() {

UnitTest obj=new UnitTest();

assertEquals(24,obj.add(12,12));

}

@Test

public void testIncorrect() {

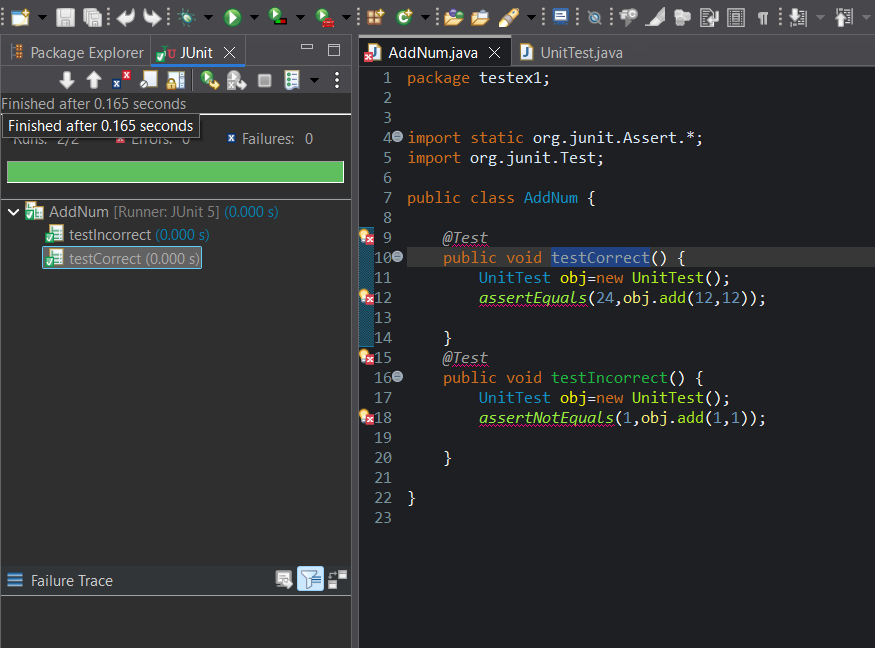
UnitTest obj=new UnitTest();

assertNotEquals(1,obj.add(1,1));

}

}

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.**

**Solution Code:**

**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());**

**}**

**} Solution:**

package ex2;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AssertCheck {

@Test

public void testCorrect() {

UnitTest obj=new UnitTest();

assertEquals(4,obj.square(2));

assertEquals(49,obj.square(7));

}

@Test

public void testNotCorrect() {

UnitTest obj=new UnitTest();

assertNotEquals(6,obj.square(2));

assertNotEquals(12,obj.square(7));

}

}

package ex2;

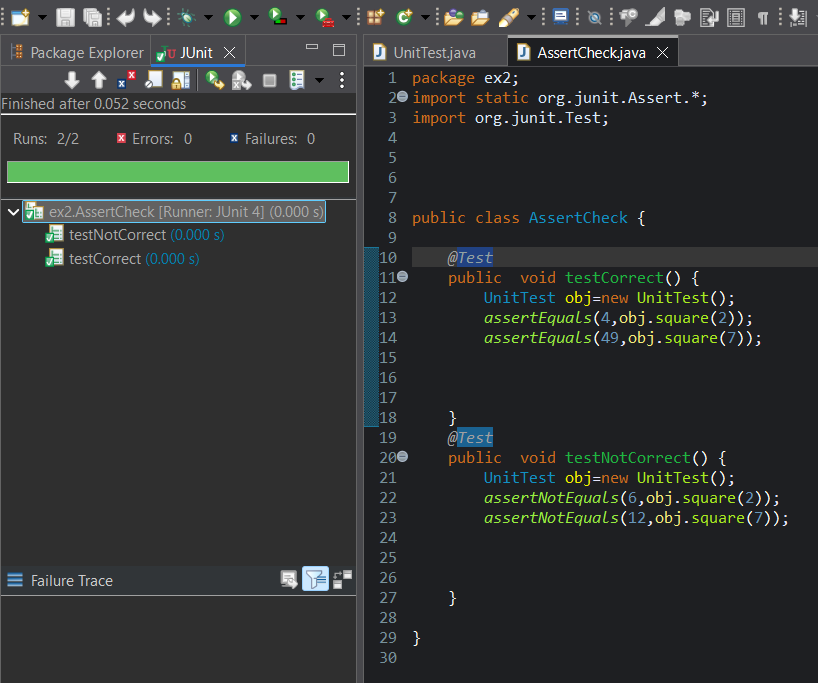
public class UnitTest {

public int square(int a) {

return a\*a;

}

}

****

**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.**

**Solution:**

package ex4;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalcTest {

private Calculator calculator;

// Setup method: runs before each test

*@Before*

public void setUp() {

calculator = new Calculator(); // Arrange

System.***out***.println("Before each test - Calculator initialized");

}

//Teardown method: runs after each test

*@After*

public void tearDown() {

calculator = null; // Clean-up

System.***out***.println("After each test - Calculator cleared");

}

*@Test*

public void testAdd() {

// Act

int result = calculator.add(10, 5);

// Assert

assertEquals(15, result);

}

private void assertEquals(int i, int result) { // **TODO** Auto-generated method stub

} // ✅ Test 2: Testing subtraction

*@Test*

public void testSubtract() {

// Act

int result = calculator.subtract(10, 5); // Assert

assertEquals(5, result);

}

}

package ex4;

public class Calculator {

public int add(int a, int b) {

return a + b;

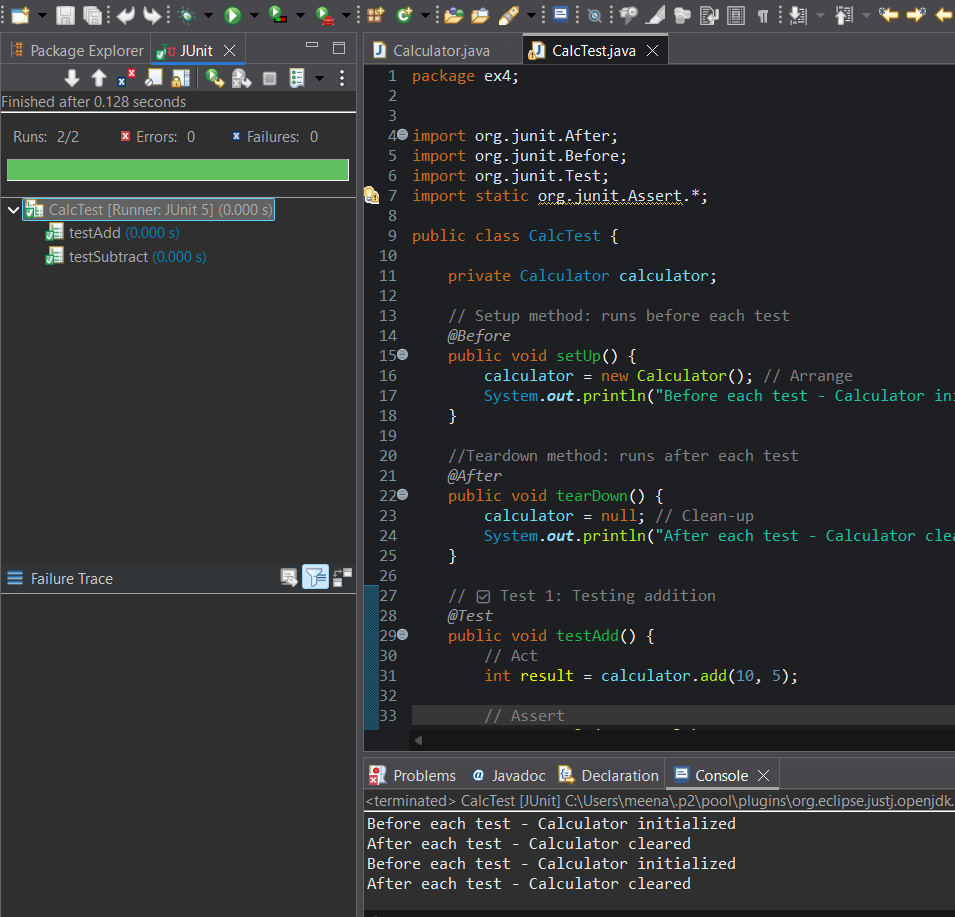
}

public int subtract(int a, int b) {

return a - b;

}

}

****