**SOLUTION WEEK 2**

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

public class App {

    public  int add(int a,int b){

        return a+b;

    }

}

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

import org.junit.jupiter.api.Test;

public class addtest {

    @Test

    void testadd(){

        App sum=new App();

        int expected=5;

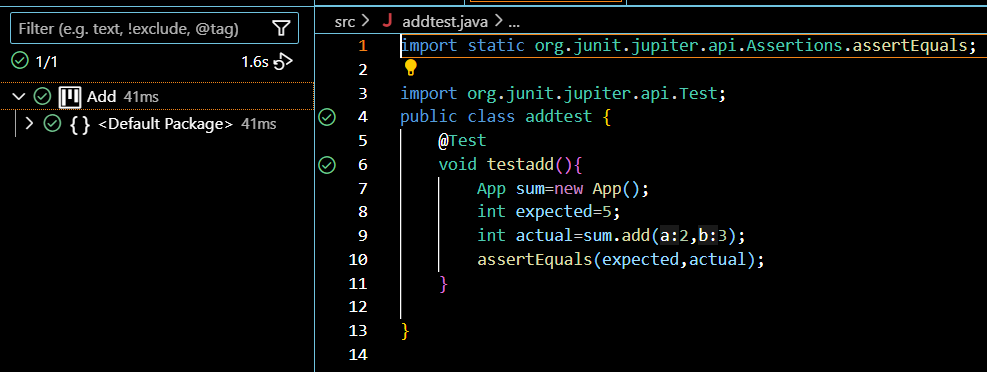
        int actual=sum.add(2,3);

    assertEquals(expected,actual);

    }

}

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());

}

}

CODE

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

import org.junit.jupiter.api.Test;

public class test{

    @Test

public void testAssertions() {

// Assert equals

assertEquals(5, 2 + 3);

}

@Test

public void test2(){

    // Assert true

assertTrue(5 > 3);

}

@Test

public void test3(){

// Assert false

assertFalse(5 < 3);

}

@Test

public void test4(){

// Assert null

assertNull(null);

}

@Test

public void test5(){

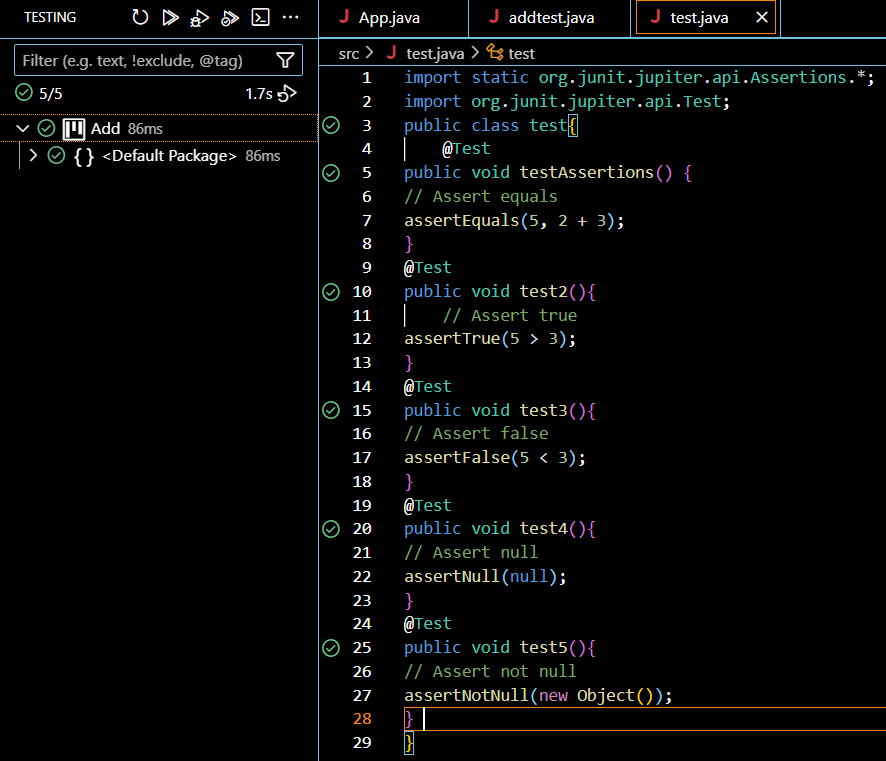
// Assert not null

assertNotNull(new Object());

}

}

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.

CODE

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

import org.junit.jupiter.api.AfterEach;

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

public class aaa {

    App res;

    @BeforeAll

    public static void Init(){

        System.out.println("Testing will start");

    }

    @BeforeEach

    void init(){

        //Arrange

         res=new App();

    }

    @AfterEach

    void cleanup(){

        System.out.println("Testing is done");

    }

    @Test

     void testadd(){

        int expected=6;

        //Act

        int actual=res.add(3,3);

        //Assert

        assertEquals(expected,actual);

    }

     @Test

     void testsub(){

  int expected=0;

        //Act

        int actual=res.sub(3,3);

        //Assert

        assertEquals(expected,actual);

    }

      @Test

     void testmul(){

        int expected=9;

        //Act

        int actual=res.mul(3,3);

        //Assert

        assertEquals(expected,actual);

    }

}

public class App {

    public  int add(int a,int b){

        return a+b;

    }

      public  int sub(int a,int b){

        return a-b;

    }

     public  int mul(int a,int b){

        return a\*b;

    }

}

OUTPUT

