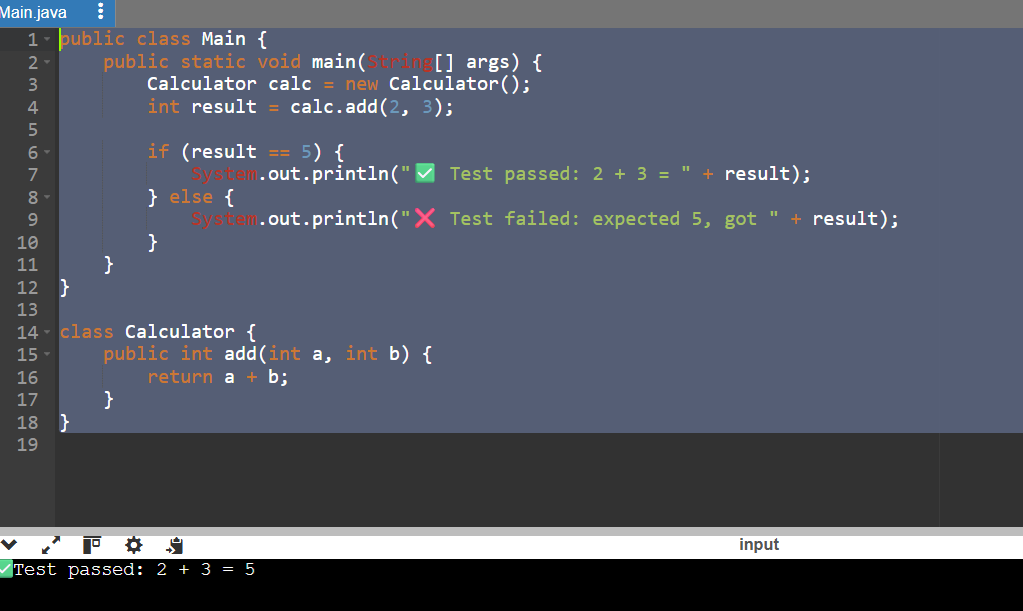
Exercise 1:

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: junit junit 4.13.2 test

3. Create a new test class in your project.



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()); } }

public class Main {

public static void main(String[] args) {

AssertionsTest test = new AssertionsTest();

test.testAssertions();

}

}

class AssertionsTest {

public void testAssertions() {

// Simulate assertEquals(5, 2 + 3)

if (2 + 3 == 5) {

System.out.println(" assertEquals passed");

} else {

System.out.println(" assertEquals failed");

}

// Simulate assertTrue(5 > 3)

if (5 > 3) {

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

} else {

System.out.println(" assertTrue failed");

}

// Simulate assertFalse(5 < 3)

if (!(5 < 3)) {

System.out.println(" assertFalse passed");

} else {

System.out.println(" assertFalse failed");

}

// Simulate assertNull(null)

Object obj1 = null;

if (obj1 == null) {

System.out.println(" assertNull passed");

} else {

System.out.println(" assertNull failed");

}

// Simulate assertNotNull(new Object())

Object obj2 = new Object();

if (obj2 != null) {

System.out.println(" assertNotNull passed");

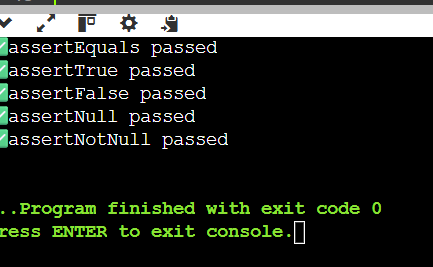
} else {

System.out.println("assertNotNull failed");

}

}

}



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

public class Main {

public static void main(String[] args) {

CalculatorTest test = new CalculatorTest();

test.setUp(); // Simulate @Before

test.testAdd(); // Run test

test.tearDown(); // Simulate @After

}

}

class Calculator {

public int add(int a, int b) {

return a + b;

}

}

class CalculatorTest {

private Calculator calc;

public void setUp() {

System.out.println(" Setup");

calc = new Calculator();

}

public void tearDown() {

System.out.println(" Teardown");

calc = null;

}

public void testAdd() {

// Arrange

int a = 2;

int b = 3;

// Act

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

// Assert

if (result == 5) {

System.out.println(" Test Passed: 2 + 3 = " + result);

} else {

System.out.println(" Test Failed: Expected 5, got " + result);

}

}

}

