WEEK 2 – ASSIGNMENT

Superset ID: 6390124

JUnit Testing Exercises:-

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.

Calculator.java

```
package com.example;

public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
}
```

CalculatorTest.java

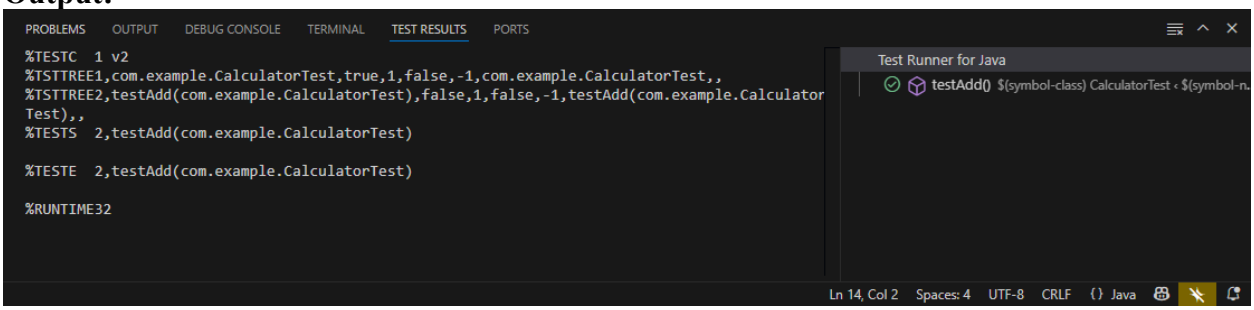
```
package com.example;

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);
    }
}
```

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());
   }
}
```

AssertionsTest.java

```
package com.example;

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

public class AssertionsTest {

    @Test
    public void testAssertions() {
        // Assert equals
        assertEquals(5, 2 + 3);
```

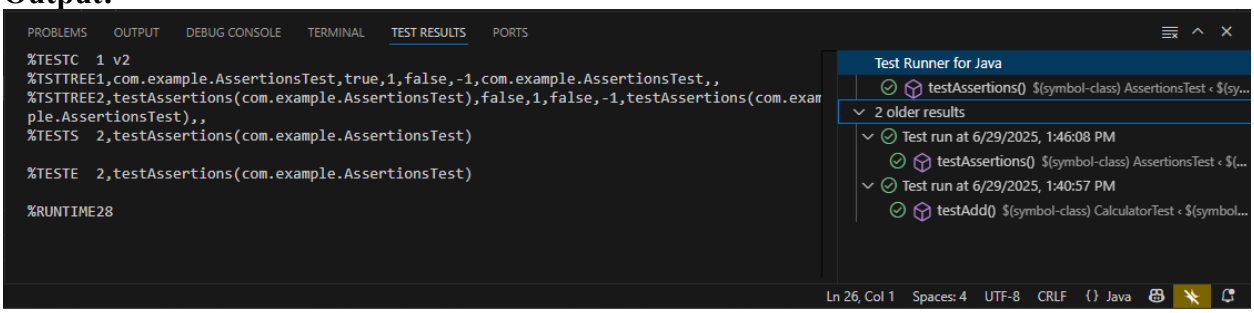
```
        // Assert true
        assertTrue(5 > 3);

        // Assert false
        assertFalse(5 < 3);

        // Assert null
        assertNull(null);

        // 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.

Calculator1.java

```
package com.example;

public class Calculator1 {
    public int add(int a, int b) {
        return a + b;
    }

    public int subtract(int a, int b) {
        return a - b;
    }
}
```

CalculatorTest1.java

```
package com.example;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
```

```java
import static org.junit.Assert.*;

public class CalculatorTest1 {

    private Calculator1 calculator;

    // Setup - runs before each test
    @Before
    public void setUp() {
        calculator = new Calculator1(); // Arrange
        System.out.println("Setup: New Calculator1 created");
    }

    // Teardown - runs after each test
    @After
    public void tearDown() {
        System.out.println("Teardown: Calculator1 test completed\n");
    }

    @Test
    public void testAdd() {
        // Act
        int result = calculator.add(10, 5);

        // Assert
        assertEquals(15, result);
    }

    @Test
    public void testSubtract() {
        // Act
        int result = calculator.subtract(10, 5);

        // Assert
        assertEquals(5, result);
    }
}
```

Output: