Exercise -1 j unit

Created	@June 25, 2025 10:04 AM
Tags	
Status	Done

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`:

```xml
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13.2</version>
  <scope>test</scope>
</dependency>
```

Result

Once you've configured Maven, you can start writing test classes using `@Tet`.

Exercise 2: Writing Basic JUnit Tests

Scenario

You need to write basic JUnit tests for a simple Java class.

Step 1: Java Class – `Calculator.java`

```
package com.example;

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

Step 2: Test Class – `CalculatorTest.java`

```
java
Copy
package com.example;

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

public class CalculatorTest {

    @Test
    public void testAdd() {
        Calculator calc = new Calculator();
        assertEquals(5, calc.add(2, 3));
    }
    @Test
    public void testSubtract() {
        Calculator calc = new Calculator();
        assertEquals(1, calc.subtract(4, 3));
    }
}
```

Output

```
Running com.example.CalculatorTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
```

Exercise 3: Assertions in JUnit

Scenario

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

Code – `AssertionsTest.java`

```
package com.example;

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

public class AssertionsTest {

    @Test
    public void testAssertions() {
        assertEquals(5, 2 + 3);
        assertTrue(5 > 3);
        assertFalse(5 < 3);
        assertNull(null);
        assertNotNull(new Object());
    }
}
```

Output

```
Running com.example.AssertionsTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```

If any assertion fails:

```
java.lang.AssertionError: expected:<6> but was:<5>
```

Exercise 4: AAA Pattern, Test Fixtures, Setup and Teardown

Scenario

You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

Code – `AdvancedCalculatorTest.java`

```
package com.example;

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

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

public class AdvancedCalculatorTest {

    private Calculator calc;

    @Before
    public void setUp() {
        calc = new Calculator();
        System.out.println("Setup complete.");
    }

    @After
    public void tearDown() {
        calc = null;
        System.out.println("Teardown complete.");
    }

    @Test
    public void testAdditionUsingAAA() {
        int a = 10, b = 15;
        int result = calc.add(a, b);
        assertEquals(25, result);
    }

    @Test
    public void testSubtractionUsingAAA() {
        int a = 20, b = 5;
        int result = calc.subtract(a, b);
        assertEquals(15, result);
    }
}
```

Output

```
Setup complete.
Teardown complete.
Setup complete.
Teardown complete.
Running com.example.AdvancedCalculatorTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
```