**Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

**Arrange–Act–Assert (AAA) Pattern**

|  |
| --- |
| **Arrange** |

|  |  |
| --- | --- |
| Set up test data, mocks, dependencies | |
| **Act** |

|  |  |
| --- | --- |
| Call the method under test | |
| **Assert** |

|  |
| --- |
| Verify that the result is as expected |

**Example of AAA in JUnit 5**

@Test

void testAddition() {

// Arrange

Calculator calc = new Calculator();

int a = 5, b = 7;

// Act

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

// Assert

Assertions.assertEquals(12, result);

}

**Test Fixtures**

A **test fixture** is the **fixed state of a set of objects** used as a baseline for running tests.  
It includes:

* Shared data (e.g., database connection, user object)
* Setup/teardown logic

Fixtures avoid repeating setup in each test method.

**Setup and Teardown Methods in JUnit 5**

|  |
| --- |
| @BeforeEach |

|  |  |
| --- | --- |
| Run **before every test method** | |
| @AfterEach |

|  |  |
| --- | --- |
| Run **after every test method** | |
| @BeforeAll |

|  |  |
| --- | --- |
| Run **once before all tests** (static) | |
| @AfterAll |

|  |
| --- |
| Run **once after all tests** (static) |

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

class Calculator {

int add(int a, int b) {

return a + b;

}

int subtract(int a, int b) {

return a - b;

}

}

public class CalculatorTest {

Calculator calc;

@BeforeEach

void setUp() {

// Arrange (common setup)

calc = new Calculator();

}

@AfterEach

void tearDown() {

// Clean-up if needed

calc = null;

}

@Test

void testAddition() {

// Act

int result = calc.add(10, 20);

// Assert

Assertions.assertEquals(30, result);

}

@Test

void testSubtraction() {

// Act

int result = calc.subtract(20, 5);

// Assert

Assertions.assertEquals(15, result);

}

}