superset id: 6405870

Unit Testing in Java Using the AAA Pattern

Objective:
To demonstrate the use of the AAA (Arrange-Act-Assert) pattern in Java unit testing using JUnit 5 by testing a simple Multiplier class.

1. Introduction to AAA Pattern
The AAA pattern is a widely adopted structure for writing unit tests. It stands for:

- **Arrange**: Prepare the necessary preconditions and inputs.
- **Act**: Perform the actual work or call the method to be tested.
- **Assert**: Verify that the outcome is as expected.

This approach improves test readability, organization, and maintainability.

2. Multiplier.java

```
public class Multiplier {
    public int multiply(int a, int b) {
        return a * b;
    }

    public int square(int a) {
        return a * a;
    }
}
```

This class contains two simple mathematical methods:

- multiply(int a, int b): Returns the product of two integers.
- square(int a): Returns the square of a single integer.

3. MultiplierTest.java

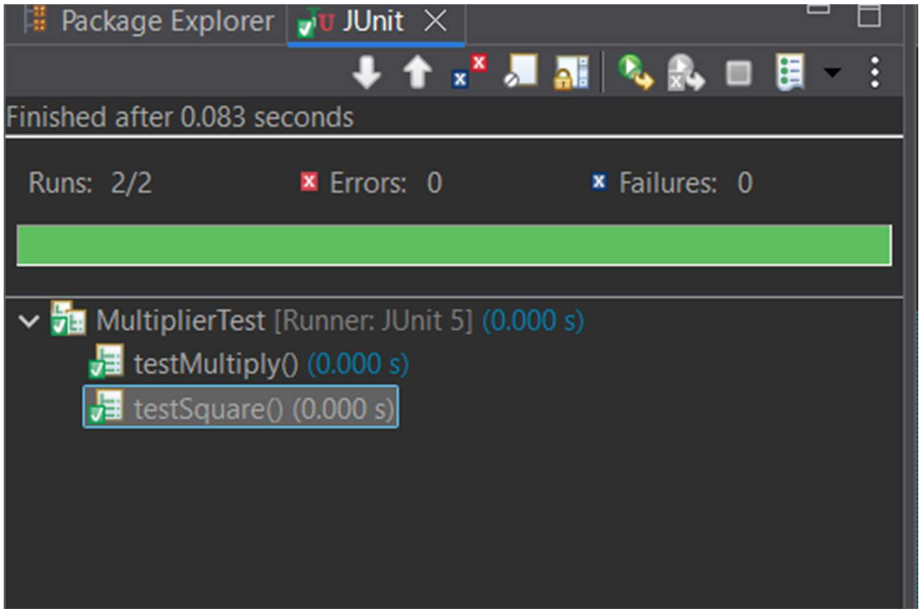
```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class MultiplierTest {

    @Test
    public void testMultiply() {
        // Arrange
        Multiplier multiplier = new Multiplier();
        int a = 4;
        int b = 3;
        int expected = 12;

        // Act
        int result = multiplier.multiply(a, b);

        // Assert
        assertEquals(expected, result, "4 multiplied by 3 should be 12");
    }
    @Test
    public void testSquare() {
        // Arrange
        Multiplier multiplier = new Multiplier();
        int a = 5;
        int expected = 26; // Incorrect value intentionally for demonstration

        // Act
        int result = multiplier.square(a);

        // Assert
        assertEquals(expected, result, "Square of 5 should be 25");
    }
}
```

---

4. Explanation of Test Cases

- **testMultiply()**
    - Arrange: Inputs 4 and 3, expected result 12
    - Act: Calls multiply(4, 3)
    - Assert: Compares the result with 12. This test will pass.
- **testSquare()**
    - Arrange: Input 5, expected result mistakenly set to 26
    - Act: Calls square(5) which returns 25
    - Assert: Compares 25 with 26. This test will fail.

This failure highlights the role of assertions and how incorrect expectations are flagged during test execution.



```
Finished after 0.083 seconds

Runs:  2/2          Errors:  0          Failures:  0

MultiplierTest [Runner: JUnit 5] (0.000 s)
    testMultiply() (0.000 s)
    testSquare() (0.000 s)
```