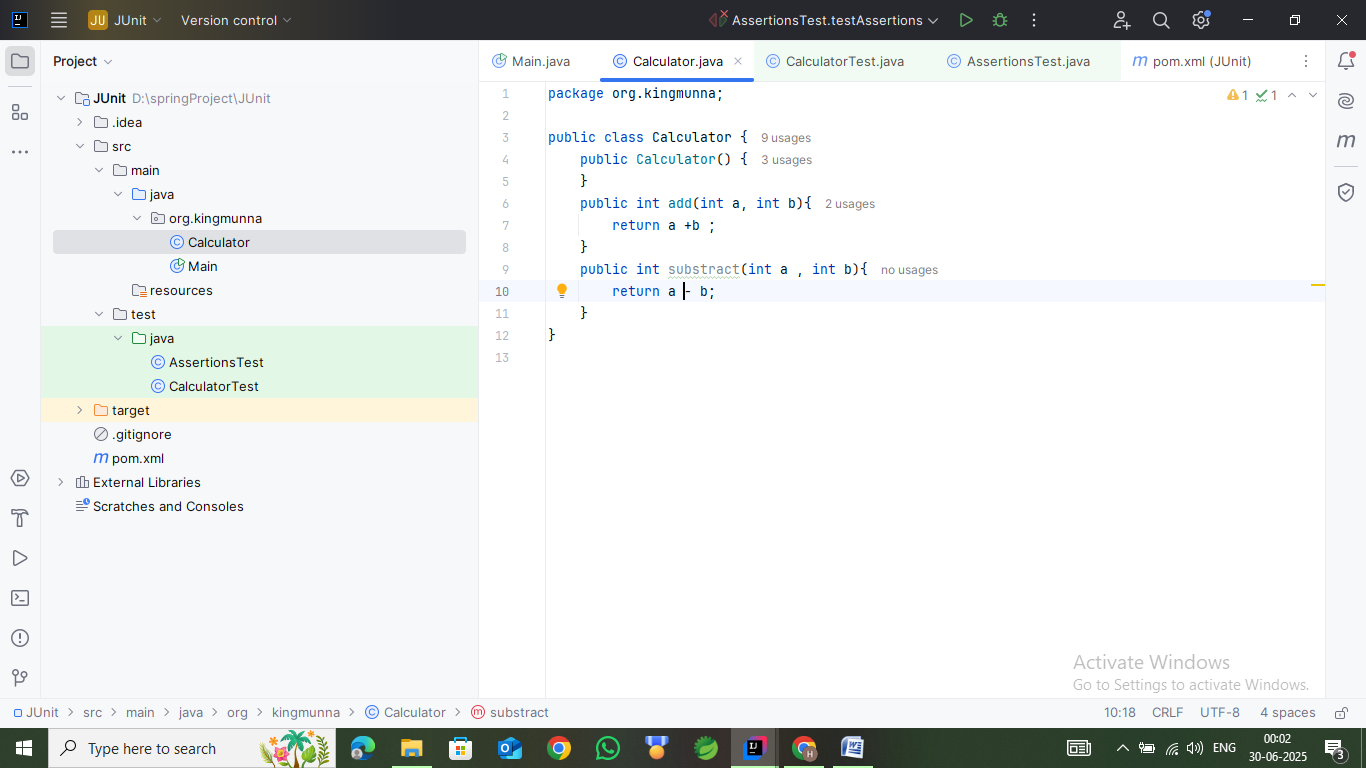
| ***Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit*** |
| --- |

The Arrange-Act-Assert (AAA) pattern is a widely used structure for writing clean and readable test methods. It divides the test into three logical parts:

* **Arrange**: Prepare all necessary objects, inputs, or dependencies required to perform the test.
* **Act**: Call the method or perform the operation that needs to be tested.
* **Assert**: Verify that the output or state is as expected.

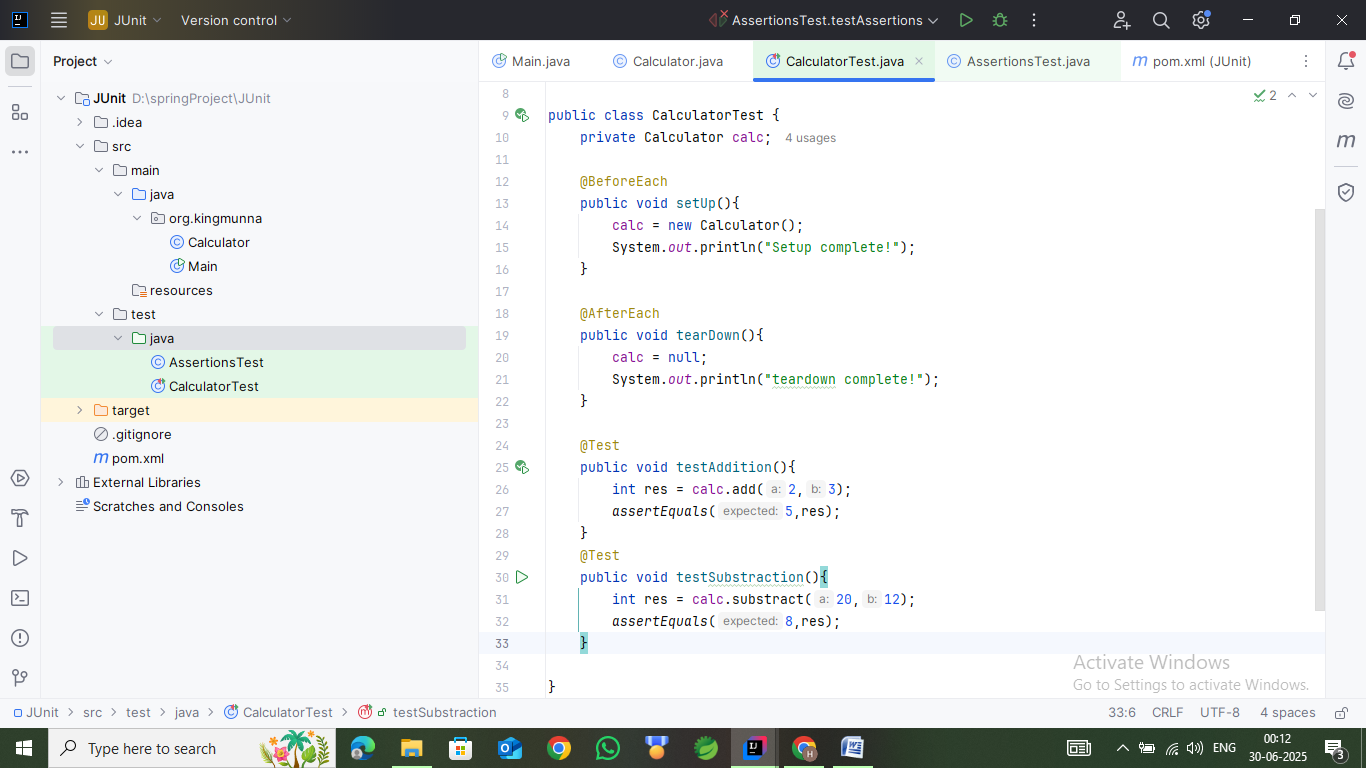
### **1. Create a class under test**

Before writing the test, create a simple class whose methods will be tested. For example, In our case we just update the Calculator class.



### **2. Write test methods using the AAA Pattern**

Now, create a JUnit test class, e.g., CalculatorTest. Inside it, write test methods that clearly separate the Arrange, Act, and Assert steps.



Each method follows the AAA structure to clearly show the steps involved in the test logic.

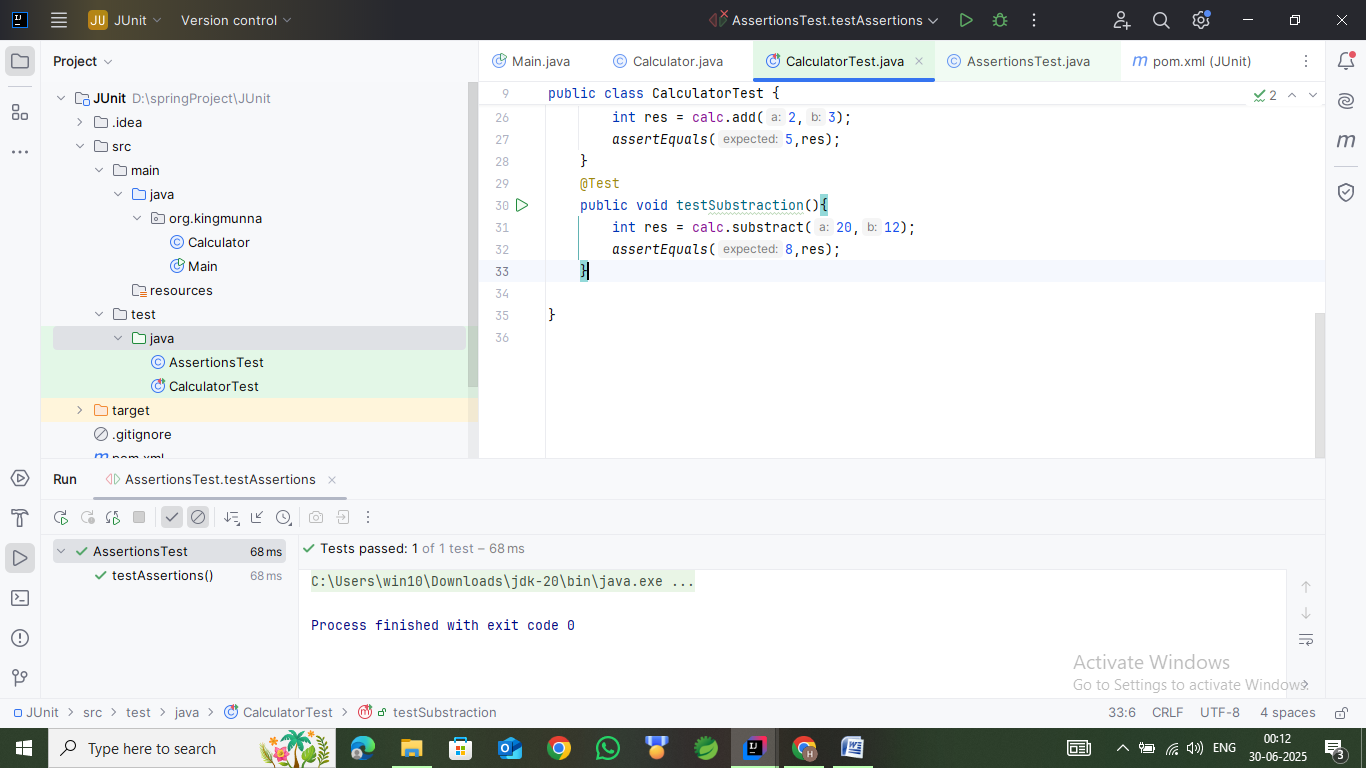
### **3. Use @Before and @After for Setup and Teardown**

In JUnit 4, @Before is used to annotate a method that should run **before each test case**, usually for setting up objects or test data.  
 @After is used to annotate a method that runs **after each test case**, often used for cleanup.

This is useful when you are reusing the same object across multiple tests, which avoids redundant code.

In this example:

* setUp() initializes the Calculator object before every test.
* tearDown() nullifies the reference after every test, simulating cleanup.



### **4. Execution Sequence**

When JUnit runs this class:

* First, it calls @Before to prepare the test fixture.
* Then, it runs the test method (like testAddition).
* Finally, it calls @After to tear down the fixture.

This happens **independently** for each test method, ensuring isolation between tests and preventing shared state bugs.