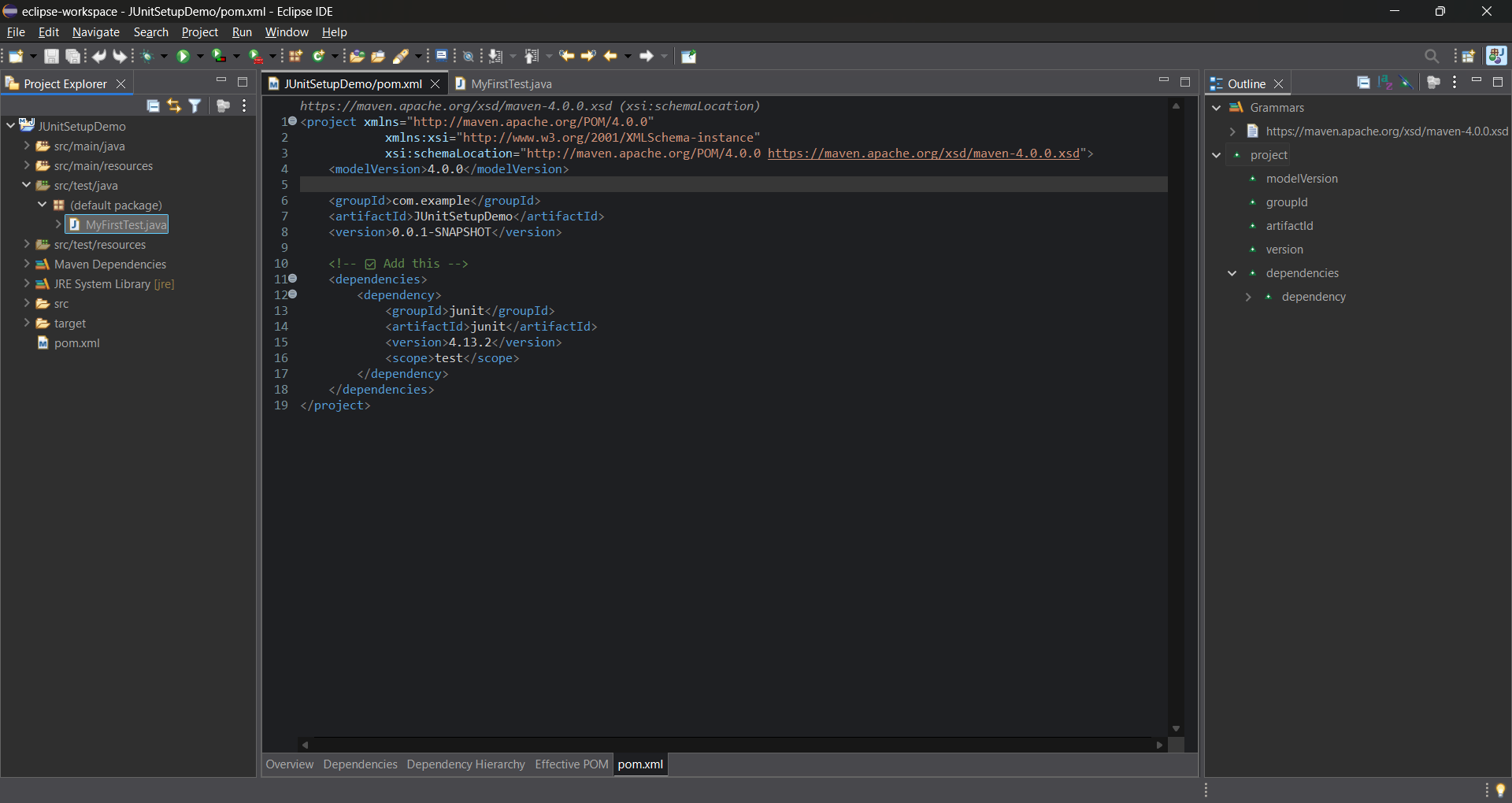
**TDD using JUnit5**

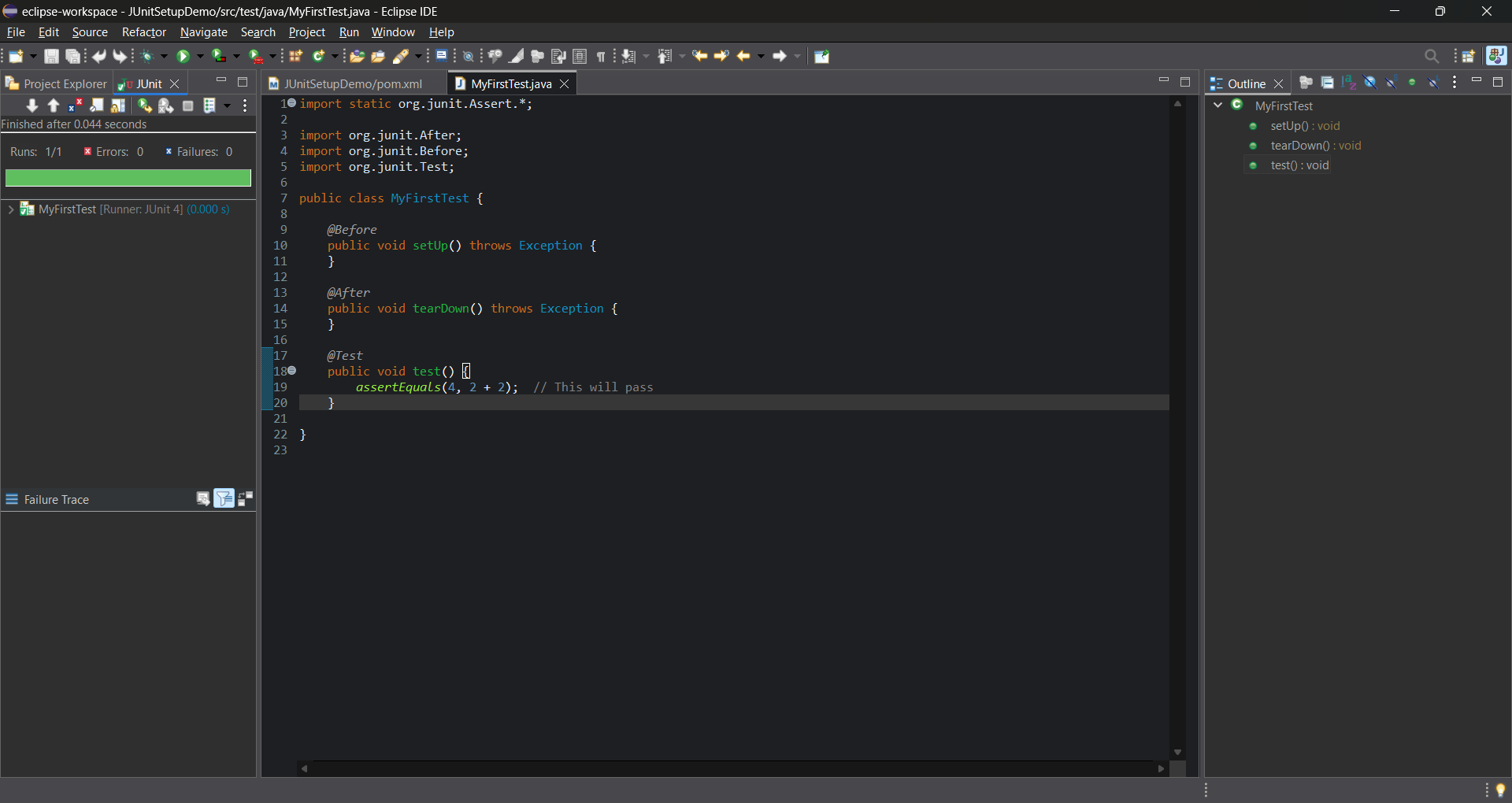
**Exercise 1: Setting Up JUnit Scenario:**

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:   
3. Create a new test class in your project.



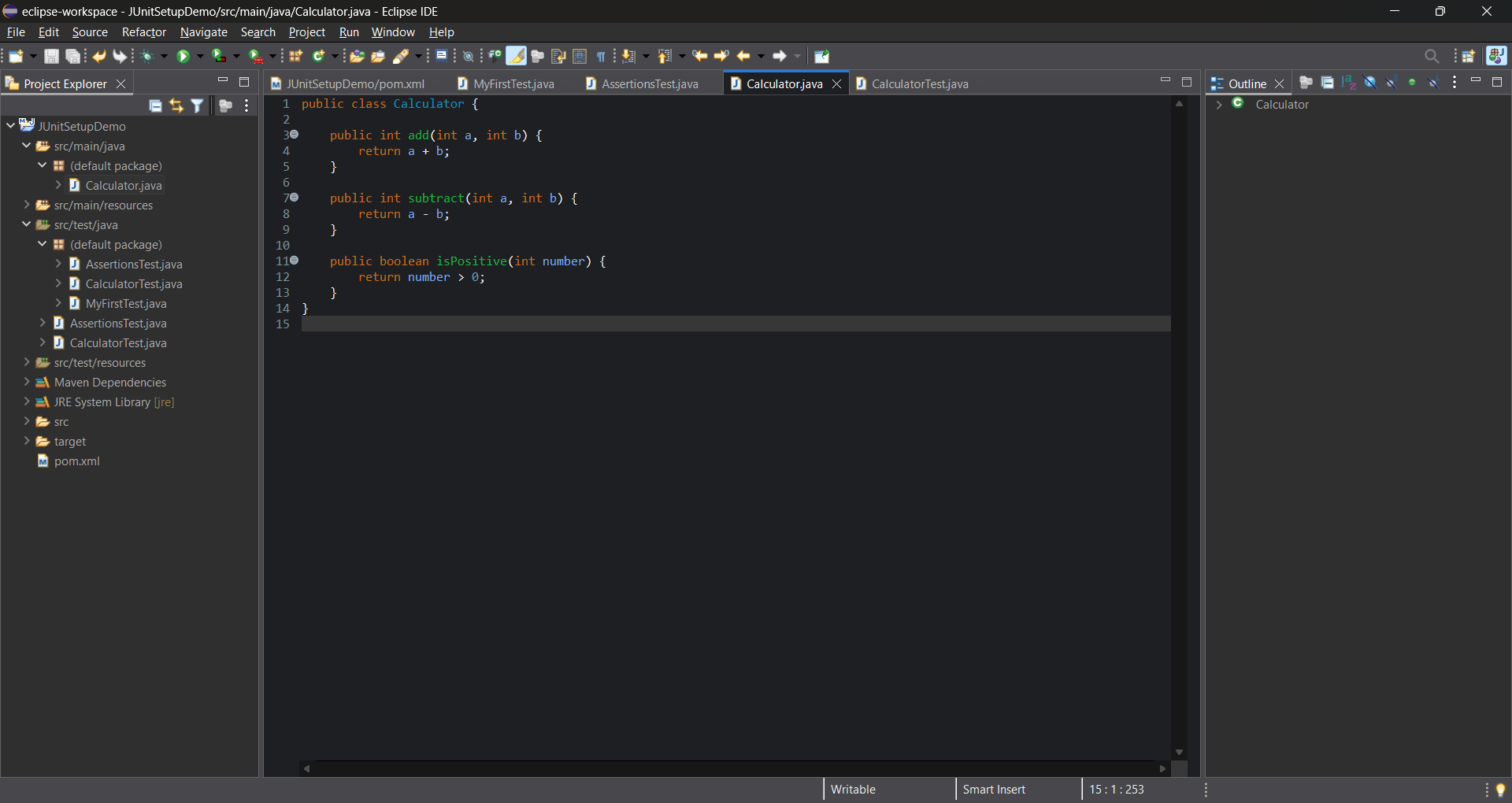


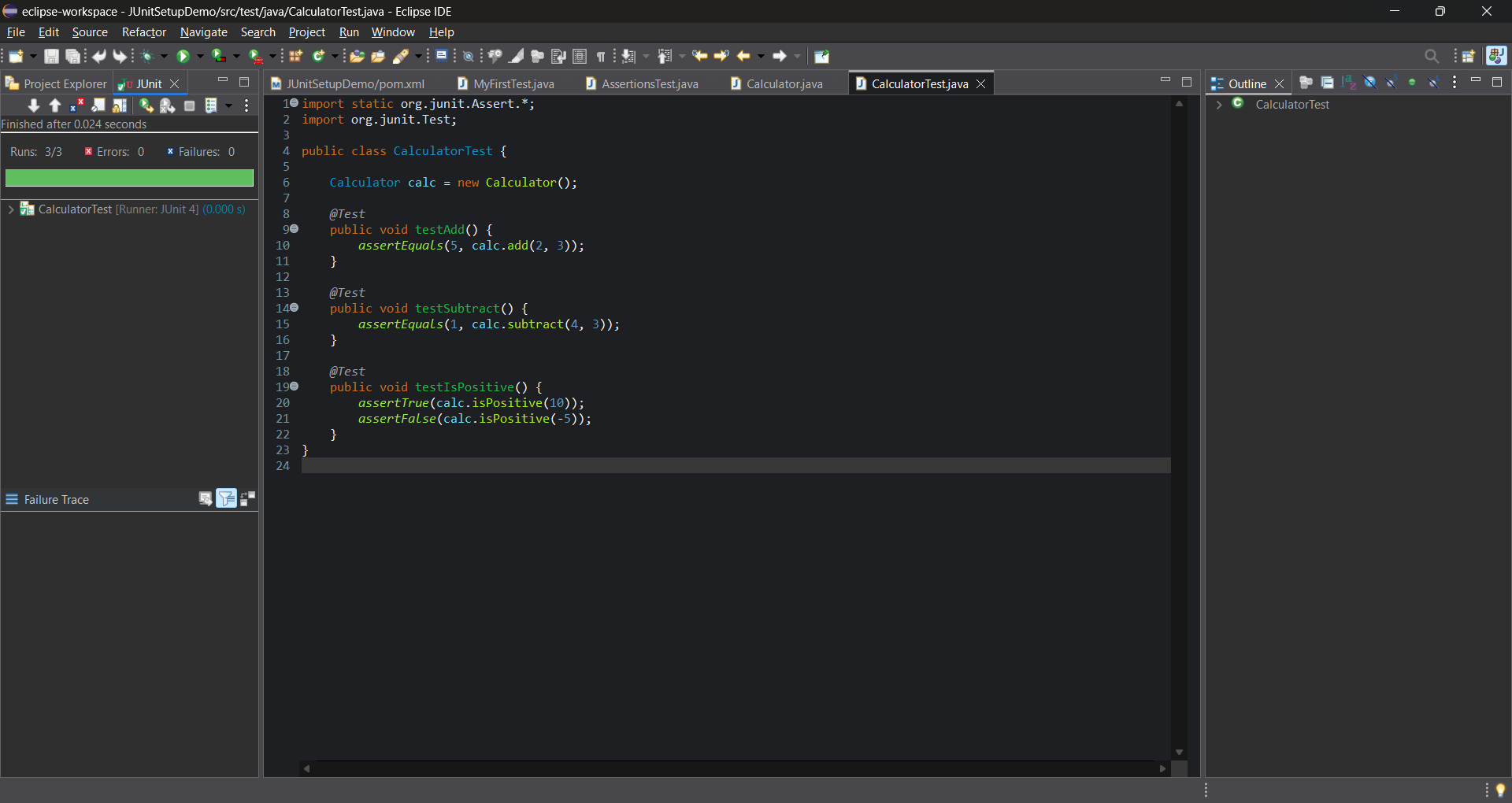
**Exercise 2:** **Writing Basic JUnit Tests Scenario:**

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

**Steps:**

1. Create a new Java class with some methods to test.   
2. Write JUnit tests for these methods.

****

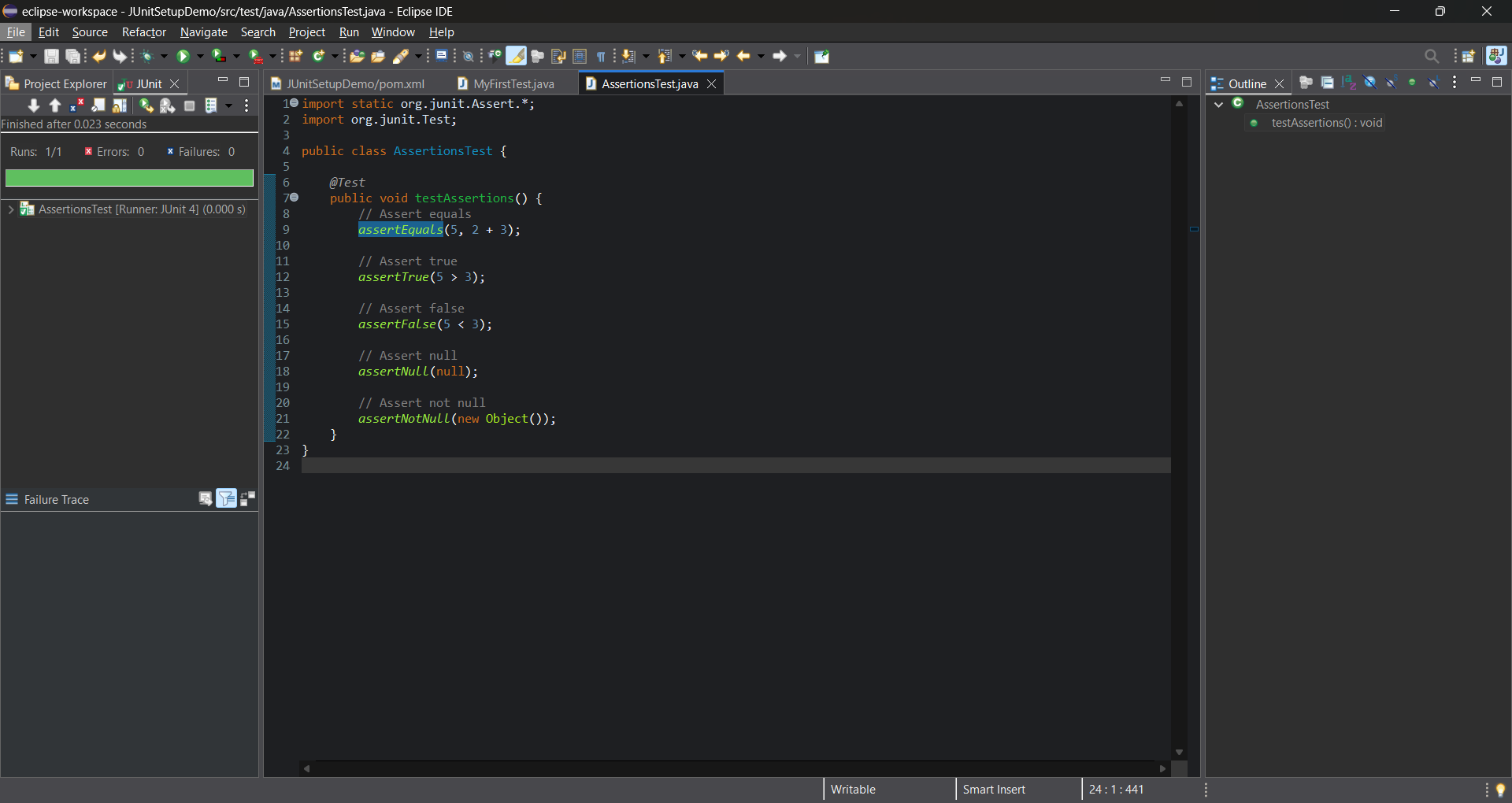
****

**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.



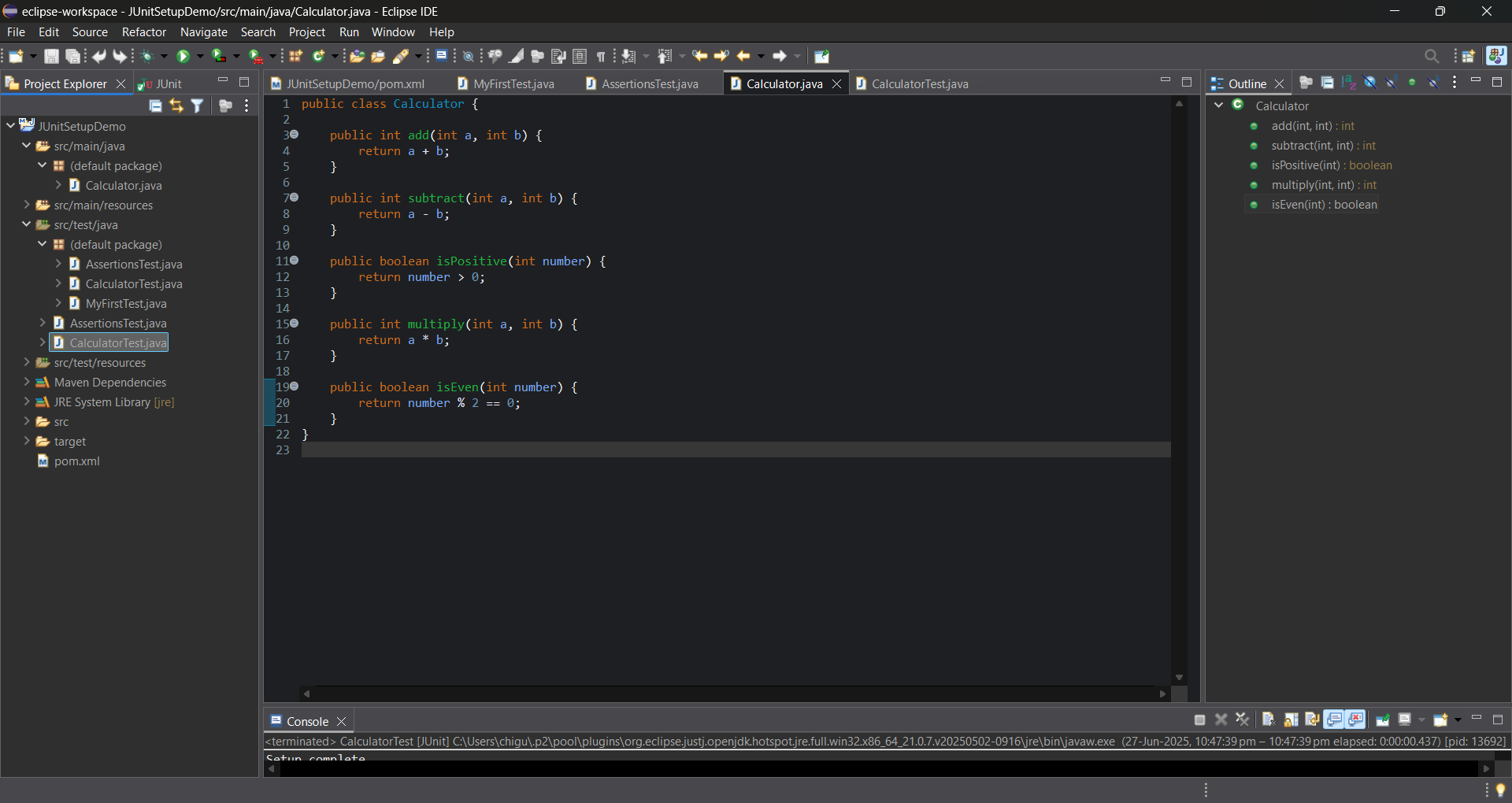
**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

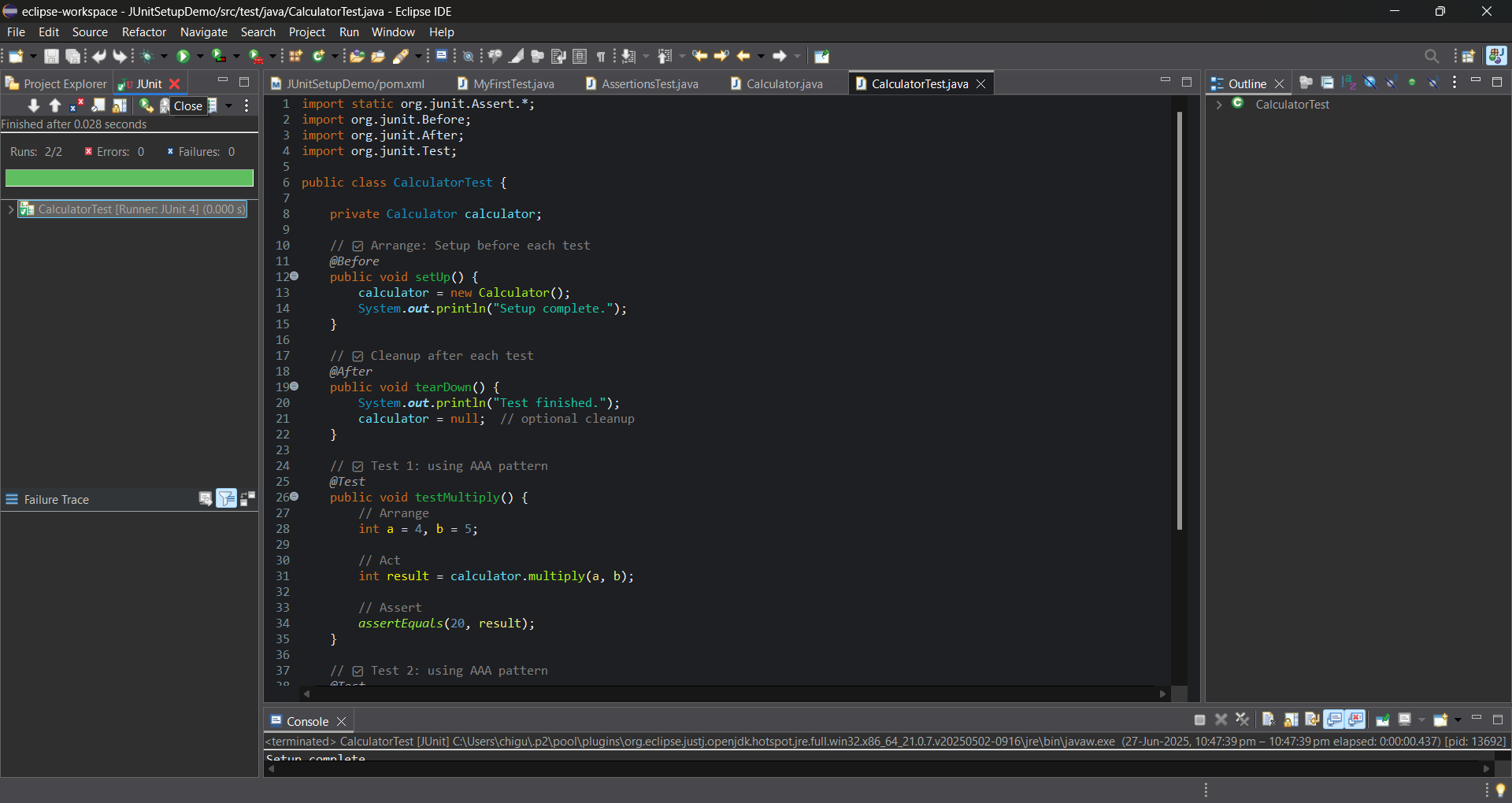
**Scenario:**

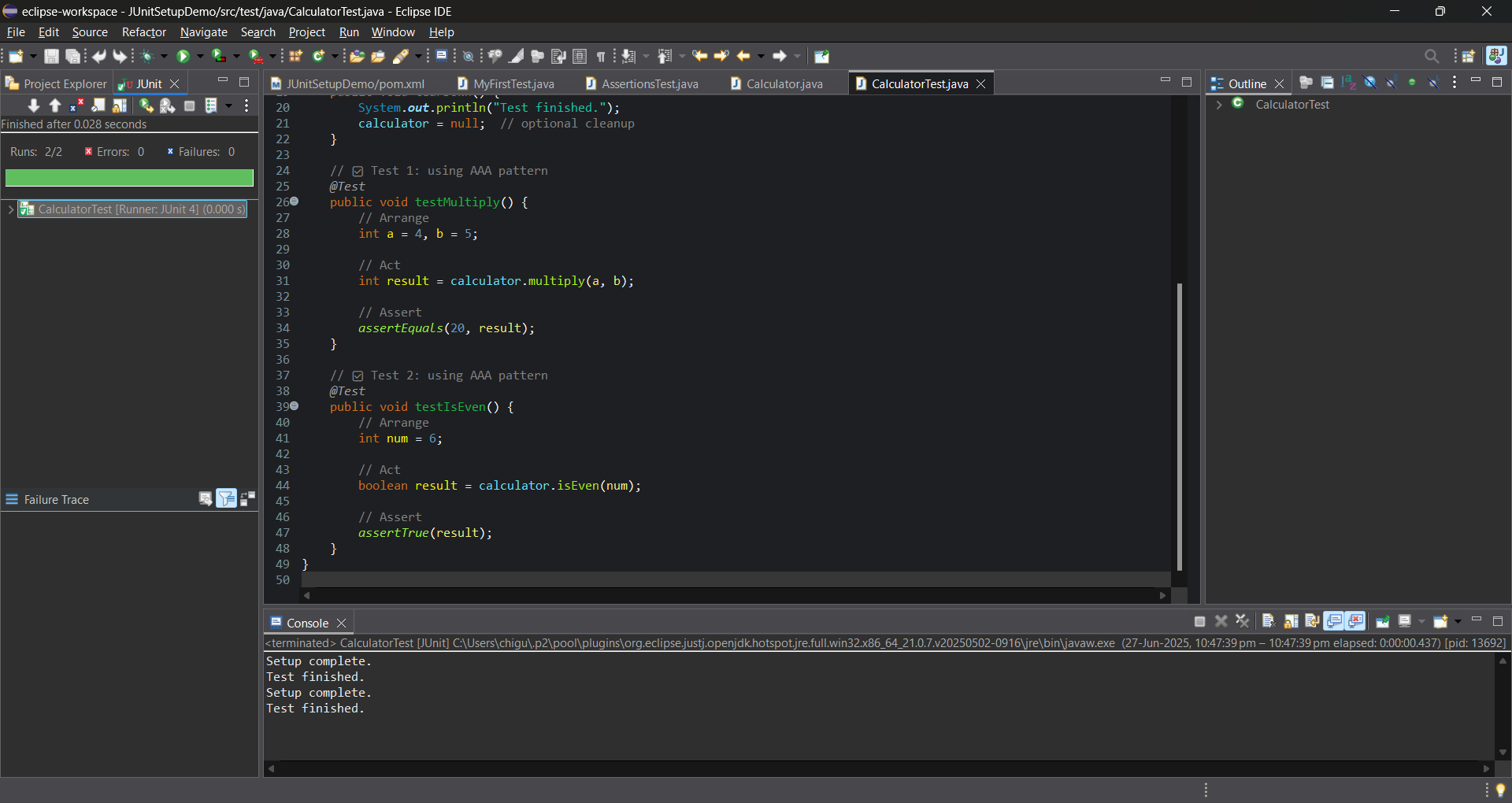
You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

**Steps:**

1. Write tests using the AAA pattern.   
2. Use @Before and @After annotations for setup and teardown methods.







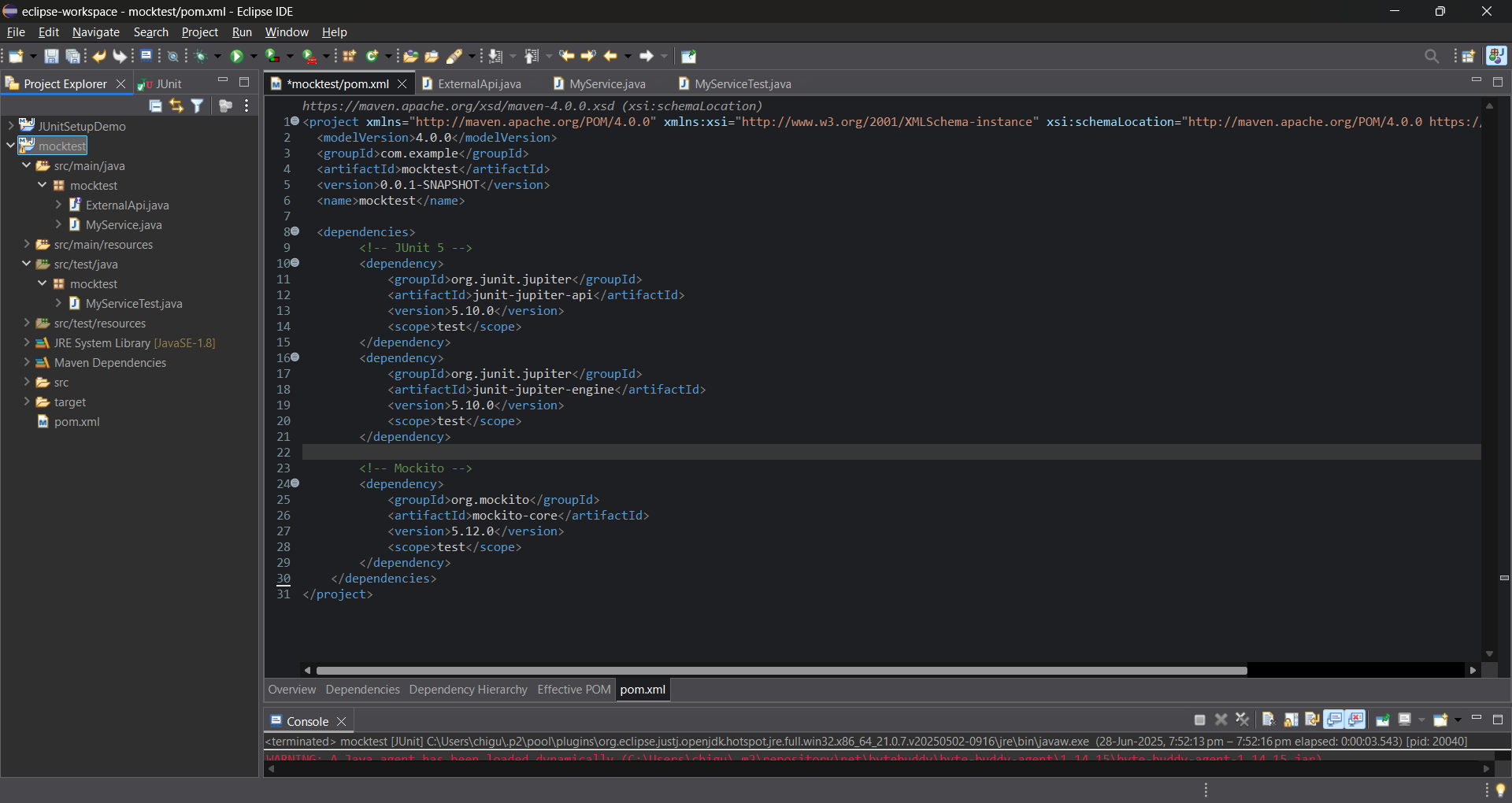
**Mockito**

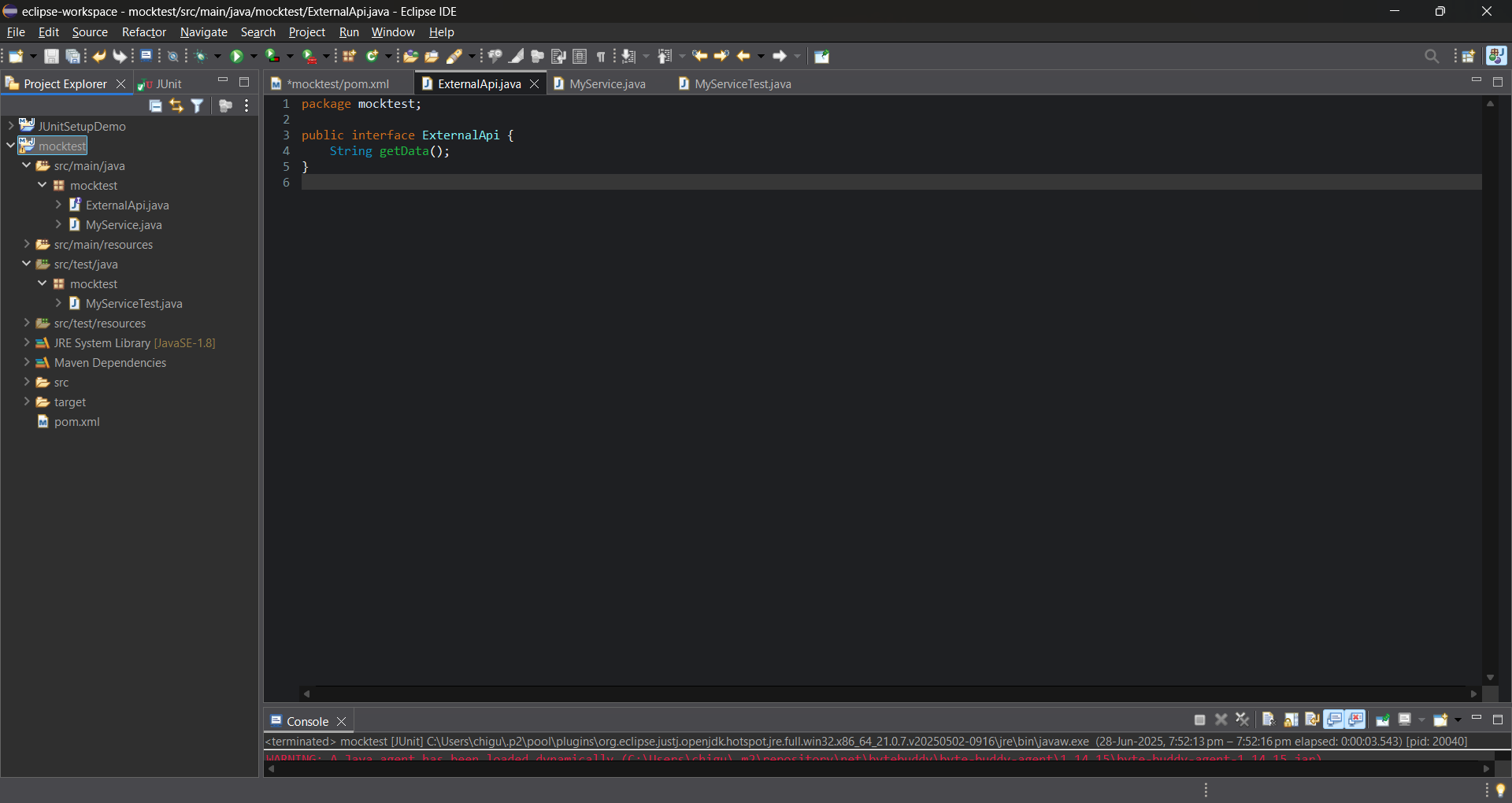
**Exercise 1:** **Mocking and Stubbing Scenario:**

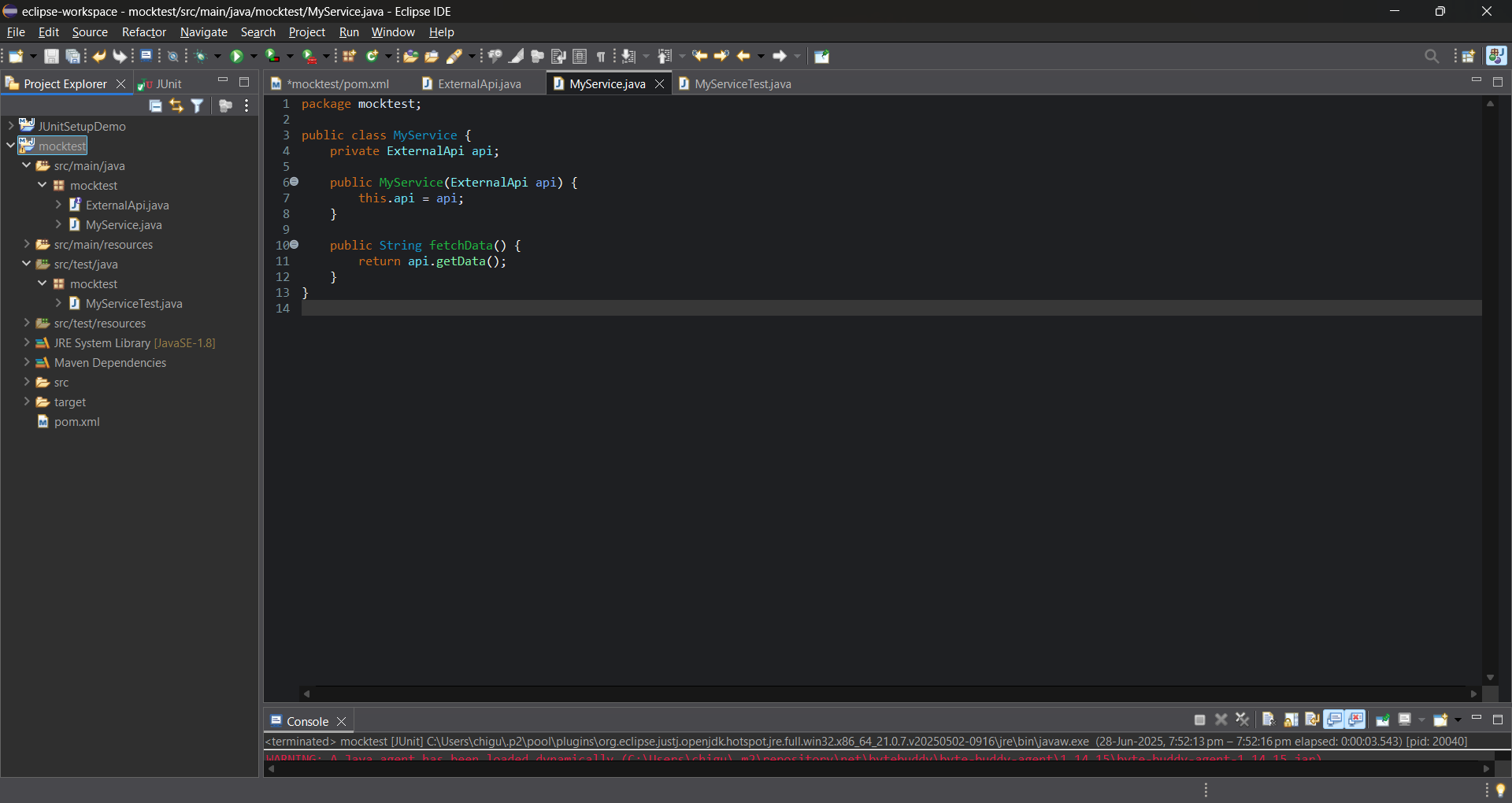
You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.

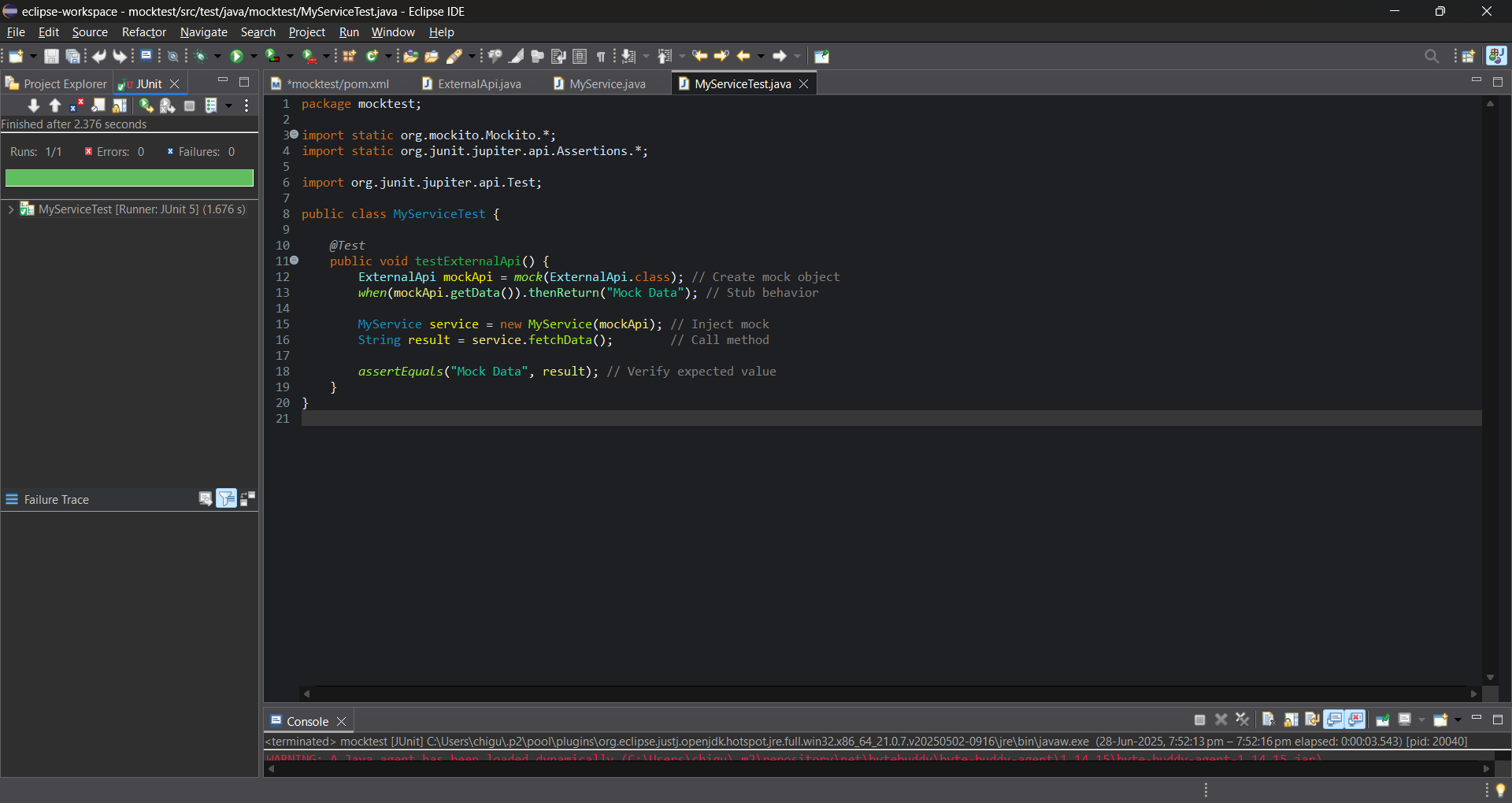
**Steps:**

1. Create a mock object for the external API.   
2. Stub the methods to return predefined values.   
3. Write a test case that uses the mock object.









**Exercise 2: Verifying Interactions Scenario:**

You need to ensure that a method is called with specific arguments.

**Steps:**

1. Create a mock object.   
2. Call the method with specific arguments.   
3. Verify the interaction.

