**Exercise 1: Setting Up JUnit**

**Scenario:**

You need to set up JUnit in your Java project to start writing unit tests.

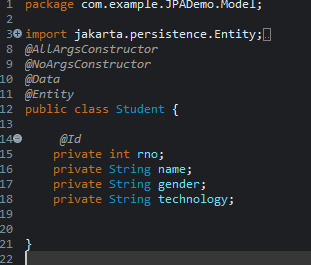
**Steps:**

1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).

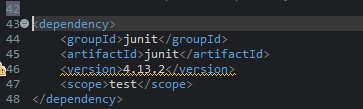
2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml:

3. Create a new test class in your project.

**STUDENT class:**



**Dependency Added in pom.xml file:**

****

**Exercise 2: Writing Basic JUnit Tests**

Scenario:

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

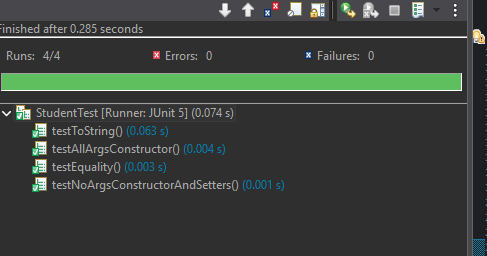
1. Create a new Java class with some methods to test.

2. Write JUnit tests for these methods

**Basic junit Test:**

****

**Result:**

****

**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..





**@Before - Setup Method**

* Method: setUp()
* Purpose: This method runs **before every test**.
* It initializes the Student object and prints a message ("🔧 Setting up Student instance...").

Ensures every test starts with a fresh Student object.

**@After - Teardown Method**

* Method: tearDown()
* Purpose: This method runs **after every test**.
* It sets the student reference to null and prints "🧹 Cleaning up Student instance...".
* Used to **release or reset** resources after each test run.

**Test Method 1: testAllArgsConstructor()**

* Uses the **all-arguments constructor** to create a Student with values.
* **Arrange**: Creates a new Student object with predefined values.
* **Act & Assert**:
  + Verifies each field using assertEquals().
  + Confirms constructor assigns values correctly.

**Test Method 2: testSettersAndGetters()**

* Uses **setters** to assign values to a Student object.
* **Arrange**: Sets roll number, name, gender, and technology.
* **Act & Assert**:
  + Confirms that the get methods return the correct values set earlier.
  + Tests the functionality of setters/getters.

**AAA Pattern**

* All tests follow the **Arrange → Act → Assert** structure:
  + **Arrange**: Set up data.
  + **Act**: Call methods or manipulate the object.
  + **Assert**: Check if the result matches expectations.