# Exercise 3 – Assertions in JUnit

## 1. Objective

Use different JUnit assertions to validate test results.

## 2. Problem Statement / Scenario

You need to write JUnit tests that employ a variety of assertions in order to demonstrate how each assertion works.

## 3. Approach / Steps

3.1 Understanding JUnit

JUnit is a popular testing framework for Java that allows developers to write and run repeatable unit tests. The latest version, JUnit 5 (a.k.a. Jupiter), introduces more flexible architecture and more readable annotations.

3.2 Setup

Ensure JUnit 5 is added to your project’s build system (e.g., Maven or Gradle). Below is a Maven dependency example:

<dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter</artifactId>  
 <version>5.11.0</version>  
 <scope>test</scope>  
</dependency>

3.3 Implementation

## 4. Code

import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class AssertionsTest {  
 @Test  
 public void testAssertions() {  
 // Assert equals  
 assertEquals(5, 2 + 3, "2 + 3 should equal 5");  
  
 // Assert true  
 assertTrue(5 > 3, "5 is greater than 3");  
  
 // Assert false  
 assertFalse(5 < 3, "5 is not less than 3");  
  
 // Assert null  
 assertNull(null, "Value should be null");  
  
 // Assert not null  
 assertNotNull(new Object(), "Object should not be null");  
 }  
}

## 5. Output Verification

When you run the above test class, you should see output similar to the following in your test runner:

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
BUILD SUCCESS

## 6. Conclusion

By using various JUnit assertions, you can thoroughly verify different aspects of your code's behavior and ensure quality.