TestNG Assertions Guide

✅ TestNG Assertions – Full Guide with Examples

In **TestNG**, **assertions** are used to **verify expected vs. actual results** in automated tests. If an assertion fails, the test case is marked as **FAILED**. Assertions help confirm that your application behaves as expected.

🔰 Types of TestNG Assertions

TestNG provides assertions via the **`org.testng.Assert`** class. The most commonly used assertions are:

Assertion Method	Description
`assertEquals(a, b)`	Asserts that **`a == b`**
`assertNotEquals(a, b)`	Asserts that **`a != b`**
`assertTrue(condition)`	Asserts that condition is **`true`**
`assertFalse(condition)`	Asserts that condition is **`false`**
`assertNull(object)`	Asserts that the object is **`null`**
`assertNotNull(object)`	Asserts that the object is **not `null`**
`fail()`	Marks the test as failed unconditionally

You can also add a **custom message** for better reporting.

💡 Import Statement

```java
import org.testng.Assert;
```

✅ Examples of Each Assertion

1. `assertEquals`

```java
@Test
public void testEqual() {
    int expected = 5;
    int actual = 2 + 3;
    Assert.assertEquals(actual, expected, "Values are not equal!");
}
```

---

2. `assertNotEquals`

```java
@Test
public void testNotEqual() {
    String actual = "TestNG";
    Assert.assertNotEquals(actual, "JUnit", "Strings should not match!");
}
```

---

3. `assertTrue`

```java
@Test
public void testTrue() {
    boolean status = 10 > 5;
    Assert.assertTrue(status, "Condition is false!");
}
```

---

4. `assertFalse`

```java
@Test
public void testFalse() {
    boolean status = "apple".contains("z");
    Assert.assertFalse(status, "Condition is true!");
}
```

---

5. `assertNull`

```java
@Test
public void testNull() {
    String str = null;
    Assert.assertNull(str, "String is not null!");
}
```

---

6. `assertNotNull`

```java
@Test
public void testNotNull() {
    String str = "TestNG";
    Assert.assertNotNull(str, "String is null!");
}
```

---

7. `fail()` — Force fail a test

```java
@Test
public void testForceFail() {
    if (true) {
        Assert.fail("Failing test intentionally!");
    }
}
```

🧪 Sample Output on Assertion Failure

If an assertion fails, TestNG will show:

```less
java.lang.AssertionError: Values are not equal! expected [5] but found [4]
```

🧠 Best Practices

- Use meaningful custom messages in assertions.
- Don't overuse assertions – validate only what's needed.
- Use `assertTrue` and `assertFalse` for boolean logic checks.
- Use `assertEquals` for comparing actual vs. expected results.

Would you like a **practical mini-project** or **lab** using TestNG assertions?

Great question! In **TestNG**, assertions are classified into two types:

🔍 Hard Assertions vs Soft Assertions in TestNG

Feature	Hard Assertion	Soft Assertion
Definition	Stops execution immediately if assertion fails	Continues execution even if assertion fails
Class Used	`org.testng.Assert`	`org.testng.asserts.SoftAssert`
Test Flow	Halts test method at failure point	Collects all failures and reports them at the end
Use Case	Critical checks	Non-blocking checks
Needs `.assertAll()`?	❌ No	✅ Yes, mandatory at the end

✅ Hard Assertion Example

```java
import org.testng.Assert;
import org.testng.annotations.Test;

public class HardAssertExample {

    @Test
    public void hardAssertTest() {
        System.out.println("Step 1");
        Assert.assertEquals(1, 2, "Hard Assert failed"); // Test stops here
        System.out.println("Step 2"); // Will NOT be executed
    }
}
```

✅ Soft Assertion Example

```java
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class SoftAssertExample {

    @Test
    public void softAssertTest() {
        SoftAssert softAssert = new SoftAssert();

        System.out.println("Step 1");
        softAssert.assertEquals(1, 2, "Soft Assert failed at Step 1");

        System.out.println("Step 2");
        softAssert.assertTrue(false, "Soft Assert failed at Step 2");

        System.out.println("Step 3");
        softAssert.assertNotEquals("Hello", "Hello", "Soft Assert failed at Step 3");

        softAssert.assertAll(); // Mandatory: triggers test failure if any above failed
    }
}
```

⚠️ Important Notes

- If you **forget** **`softAssert.assertAll()`**, TestNG **won't fail** the test, even if soft assertions failed.
- Use **`SoftAssert`** when you want to collect all errors in a test and handle them together.
- Use **`Hard Assert`** when the test should stop immediately after a failure (e.g., login failure).

🧪 When to Use What?

Scenario	Recommendation
Login verification	**Hard Assert** – critical
UI label checks, field validations	**Soft Assert** – non-critical
API contract testing	Mix (Hard for status code, Soft for body fields)

Would you like a **lab-style demo** where both Hard and Soft Assertions are used in the same test scenario?

✅ `SoftAssert` Class in TestNG – Complete Guide

The **`SoftAssert`** class in **TestNG** (package **`org.testng.asserts.SoftAssert`**) allows test execution to **continue even when an assertion fails**. It collects all assertion errors and throws them **only when `assertAll()` is called**.

🔍 Why Use SoftAssert?

In scenarios like **form field validations**, **UI checks**, or **multiple assertions in one test**, you may want to verify everything—even if some checks fail—and report all issues at once.

🧱 SoftAssert Class – Key Methods

Method	Description
`assertEquals(actual, expected)`	Checks if values are equal
`assertNotEquals(actual, expected)`	Checks if values are not equal
`assertTrue(condition)`	Checks if the condition is true
`assertFalse(condition)`	Checks if the condition is false
`assertNull(object)`	Checks if the object is null
`assertNotNull(object)`	Checks if the object is not null
`fail()`	Forces a failure
`assertAll()`	✅ **Mandatory** to call at the end to trigger all collected assertion errors

---

✅ SoftAssert Example in TestNG

```java
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class SoftAssertDemo {

    @Test
    public void testWithSoftAssert() {
        SoftAssert softAssert = new SoftAssert();

        System.out.println("Step 1: Check equality");
        softAssert.assertEquals("abc", "xyz", "Strings are not equal!");

        System.out.println("Step 2: Check boolean condition");
        softAssert.assertTrue(false, "Condition is false!");

        System.out.println("Step 3: Check null");
        String str = "TestNG";
        softAssert.assertNull(str, "String is not null!");

        System.out.println("All steps executed. Now calling assertAll()");
        softAssert.assertAll(); // 🔴 Required to report the failures
    }
}
```

🧪 Output on Failure

If any assertion fails, and **`assertAll()`** is called, you'll get:

```csharp
java.lang.AssertionError:
The following asserts failed:
    Strings are not equal!
    Condition is false!
    String is not null!
```

⚠️ Common Mistakes

1. ❌ **Forgetting `assertAll()`**
   ➜ No failure will be reported even if assertions fail!
2. ❌ **Using `SoftAssert` across multiple tests**
   ➜ Use a new **`SoftAssert`** instance in each **`@Test`** method.

📦 Import Statement

```java
import org.testng.asserts.SoftAssert;
```

✅ Real-World Use Case Example (Form Validation)

```java
@Test
public void testFormValidation() {
    SoftAssert soft = new SoftAssert();

    soft.assertEquals(getLabelText("name"), "Name*", "Name label mismatch");
    soft.assertTrue(isCheckboxPresent(), "Checkbox not present");
    soft.assertEquals(getSubmitButtonText(), "Submit Now", "Submit button text wrong");

    soft.assertAll();
}
```

Would you like a downloadable or runnable **Java project template** using **`SoftAssert`**?

Certainly! Let's break down the line:

```java
Assert.assertEquals(actual, expected, "Values are not equal!");
```

🧠 Purpose

This line is using **TestNG's Hard Assertion** to check whether two values are equal.
If they are **not equal**, the test will **fail immediately**, and the message **`"Values are not equal!"`** will be shown in the test report.

🔍 Breakdown of Components

Part	Meaning
`Assert`	Class from **`org.testng.Assert`** package – contains static assertion methods
`assertEquals(...)`	Method that checks if **`actual`** and **`expected`** values are equal
`actual`	The **actual value** produced by your code/test
`expected`	The **expected value** you want the actual result to match
`"Values are not equal!"`	Custom **error message** shown if the assertion fails

✅ Example

```java
@Test
public void testSum() {
    int actual = 2 + 3;
    int expected = 5;

    Assert.assertEquals(actual, expected, "Values are not equal!");
}
```

🔎 If Assertion Passes:

- **`actual = 5`**, **`expected = 5`**
- Test passes ✅

❌ If Assertion Fails:

```java
int actual = 2 + 2;  // actual = 4
int expected = 5;

Assert.assertEquals(actual, expected, "Values are not equal!");
```

Test fails with:

```less
java.lang.AssertionError: Values are not equal! expected [5] but found [4]
```

📝 Summary

- **`Assert.assertEquals(actual, expected, message)`** compares values.
- If **equal** → test continues.
- If **not equal** → test fails immediately with your custom message.
- Used in **critical validations** (Hard Assertion).

Would you like to see **`assertEquals()`** used for **strings, arrays, or objects** as well?