1. Introduction

In this article, we're going to explore in details the assertions available within JUnit.

Following the migrating from JUnit 4 to JUnit 5 and A Guide to JUnit 5 articles, we're now going into details about the different assertions available in JUnit 4 and JUnit 5.

We'll also highlight the enhancements made on the assertions with JUnit 5.

2. Assertions

Assertions are utility methods to support asserting conditions in tests; these methods are accessible through the *Assert* class, in JUnit 4, and the *Assertions* one, in JUnit 5.

In order to increase the readability of the test and of the assertions itself, it's always recommended to *import* statically the respective class. In this way, we can refer directly to the assertion method itself without the representing class as a prefix.

Let's start exploring the assertions available with JUnit 4.

3. Assertions in JUnit 4

In this version of the library, assertions are available for all primitive types, *Objects,* and *arrays* (either of primitives or *Objects).*

The parameters order, within the assertion, is the expected value followed by the actual value; optionally the first parameter can be a *String* message that represents the message output of the evaluated condition.

There's only one slightly different in how is defined the *assertThat* assertions, but we'll cover it later on.

Let's start with the *assertEquals* one.

3.1. *assertEquals*

The *assertEquals* assertion verifies that the expected and the actual values are equal:

```
@Test
public void whenAssertingEquality_thenEqual() {
   String expected = "Baeldung";
   String actual = "Baeldung";

   assertEquals(expected, actual);
}
```

It's also possible to specify a message to display when the assertion fails:

```
assertEquals("failure - strings are not equal", expected, actual);
```

3.2. *assertArrayEquals*

If we want to assert that two arrays are equals, we can use the *assertArrayEquals:*

```
@Test
public void whenAssertingArraysEquality_thenEqual() {
   char[] expected = {'J','u','n','i','t'};
   char[] actual = "Junit".toCharArray();

   assertArrayEquals(expected, actual);
}
```

If both arrays are *null*, the assertion will consider them equal:

```
@Test
public void givenNullArrays_whenAssertingArraysEquality_thenEqual() {
   int[] expected = null;
   int[] actual = null;

   assertArrayEquals(expected, actual);
}
```

3.3. *assertNotNull* and *assertNull*

When we want to test if an object is *null* we can use the *assertNull* assertion:

```
1    @Test
2    public void whenAssertingNull_thenTrue() {
3       Object car = null;
4
5       assertNull("The car should be null", car);
6    }
```

In the opposite way, if we want to assert that an object should not be null we can use the *assertNotNull assertion.*

3.4. *assertNotSame* and *assertSame*

With *assertNotSame*, it's possible to verify if two variables don't refer to the same object:

```
1    @Test
2    public void whenAssertingNotSameObject_thenDifferent() {
3       Object cat = new Object();
4       Object dog = new Object();
5
6       assertNotSame(cat, dog);
7    }
```

Otherwise, when we want to verify that two variables refer to the same object, we can use the *assertSame* assertion.

3.5. *assertTrue* and *assertFalse*

In case we want to verify that a certain condition is *true* or *false*, we can respectively use the *assertTrue* assertion or the *assertFalse* one:

```
1    @Test
2    public void whenAssertingConditions_thenVerified() {
3       assertTrue("5 is greater then 4", 5 > 4);
4       assertFalse("5 is not greater then 6", 5 > 6);
5    }
```

3.6. *fail*

The *fail* assertion fails a test throwing an *AssertionFailedError*. It can be used to verify that an actual exception is thrown or when we want to make a test failing during its development.

Let's see how we can use it in the first scenario:

```
@Test
public void whenCheckingExceptionMessage_thenEqual() {
   try {
      methodThatShouldThrowException();
      fail("Exception not thrown");
   } catch (UnsupportedOperationException e) {
      assertEquals("Operation Not Supported", e.getMessage());
   }
}
```

3.7. *assertThat*

The *assertThat* assertion is the only one in JUnit 4 that has a reverse order of the parameters compared to the other assertions.

In this case, the assertion has an optional failure message, the actual value, and a *Matcher* object.

Let's see how we can use this assertion to check if an array contains particular values:

```
@Test
public void testAssertThatHasItems() {
   assertThat(
     Arrays.asList("Java", "Kotlin", "Scala"),
     hasItems("Java", "Kotlin"));
}
```