What is JUnit?

[JUnit](http://junit.org/) is an open source testing framework which is used to write and run repeatable automated tests, so that we can be ensured that our code works as expected.

JUnit provides:

* Assertions for testing expected results.
* Test features for sharing common test data.
* Test suites for easily organizing and running tests.
* Graphical and textual test runners

JUnit is used to test:

* an entire object
* part of an object – a method or some interacting methods
* interaction between several objects

JUnit annotations

1. [@Test](mailto:1.@Test)

public void method :-> The Test annotation indicates that the public void method to which it is attached can be run as a test case.

1. [@Before](mailto:2.@Before)

public void method -> The Before annotation indicates that this method must be executed before each test in the class, so as to execute some preconditions necessary for the test.

1. @BeforeClass

public void method() -> The BeforeClass annotation indicates that the static method to which is attached must be executed once and before all tests in the class. That happens when the test methods share computationally expensive setup (e.g. connect to database).

1. [@After](mailto:4.@After)

public void method() -> The After annotation indicates that this method gets executed after execution of each test (e.g. reset some variables after execution of every test, delete temporary variables etc)

1. @AfterClass

public void method() -> The AfterClass annotation can be used when a method needs to be executed after executing all the tests in a JUnit Test Case class so as to clean-up the expensive set-up (e.g disconnect from a database). Attention: The method attached with this annotation (similar to BeforeClass) must be defined as static.

1. @Ignore

public void method() -> The Ignore annotation can be used when you want temporarily disable the execution of a specific test. Every method that is annotated with @Ignore won’t be executed.

Ex:

package com.javacodegeeks.junit;

import static org.junit.Assert.\*;

import java.util.\*;

import org.junit.\*;

public class AnnotationsTest {

private ArrayList testList;

@BeforeClass

public static void onceExecutedBeforeAll() {

System.out.println("@BeforeClass: onceExecutedBeforeAll");

}

@Before

public void executedBeforeEach() {

testList = new ArrayList();

System.out.println("@Before: executedBeforeEach");

}

@AfterClass

public static void onceExecutedAfterAll() {

System.out.println("@AfterClass: onceExecutedAfterAll");

}

@After

public void executedAfterEach() {

testList.clear();

System.out.println("@After: executedAfterEach");

}

@Test

public void EmptyCollection() {

assertTrue(testList.isEmpty());

System.out.println("@Test: EmptyArrayList");

}

@Test

public void OneItemCollection() {

testList.add("oneItem");

assertEquals(1, testList.size());

System.out.println("@Test: OneItemArrayList");

}

@Ignore

public void executionIgnored() {

System.out.println("@Ignore: This execution is ignored");

}

}

O/P:

@BeforeClass: onceExecutedBeforeAll

@Before: executedBeforeEach

@Test: EmptyArrayList

@After: executedAfterEach

@Before: executedBeforeEach

@Test: OneItemArrayList

@After: executedAfterEach

@AfterClass: onceExecutedAfterAll

### JUnit assertions : In this section we will present a number of assertion methods. All those methods are provided by the Assert class which extends the class java.lang.Object and they are useful for writing tests so as to detect failures. In the table below there is a more detailed explanation of the most commonly used assertion methods.

|  |  |
| --- | --- |
| **Assertion** | **Description** |
| void assertEquals([String message], expected value, actual value) | Asserts that two values are equal.  Values might be type of int, short, long, byte, char  or java.lang.Object.  The first argument is an optional String message. |
| void assertTrue([String message], boolean condition) | Asserts that a condition is true. |
| void assertFalse([String message],boolean condition) | Asserts that a condition is false. |
| void assertNotNull([String message], java.lang.Object object) | Asserts that an object is not null. |
| void assertNull([String message], java.lang.Object object) | Asserts that an object is null. |
| void assertSame([String message], java.lang.Object expected, java.lang.Object actual) | Asserts that the two objects refer to the same object. |
| void assertNotSame([String message], java.lang.Object unexpected, java.lang.Object actual) | Asserts that the two objects do not refer to the  same object. |
| void assertArrayEquals([String message], expectedArray, resultArray) | Asserts that the array expected and the resulted array are equal. The type of Array might be int, long, short, char, byte or java.lang.Object. |

package com.javacodegeeks.junit;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AssertionsTest {

@Test

public void test() {

String obj1 = "junit";

String obj2 = "junit";

String obj3 = "test";

String obj4 = "test";

String obj5 = null;

int var1 = 1;

int var2 = 2;

int[] arithmetic1 = { 1, 2, 3 };

int[] arithmetic2 = { 1, 2, 3 };

assertEquals(obj1, obj2);

assertSame(obj3, obj4);

assertNotSame(obj2, obj4);

assertNotNull(obj1);

assertNull(obj5);

assertTrue(var1 var2);

assertArrayEquals(arithmetic1, arithmetic2);

}

}

In the class above we can see how these assert methods work.

* The assertEquals() method will return normally if the two compared objects are equal, otherwise a failure will be displayed in the JUnit window and the test will abort.
* The assertSame() and assertNotSame() methods tests if two object references point to exactly the same object.
* The assertNull() and assertNotNull() methods test whether a variable is null or not null.
* The assertTrue() and assertFalse() methods tests if a condition or a variable is true or false.
* The assertArrayEquals() will compare the two arrays and if they are equal, the method will proceed without errors. Otherwise, a failure will be displayed in the JUnit window and the test will abort.