Assertions применяется в тестировании и определяет тест passed или faild (*a way to verify that the expected result and the actual result matched or not*). If we could decide the **outcome** on different small methods using assertions in our test case, we can determine whether our test failed or passed overall. An example of assertion can be logging into the website, checking the title of the webpage, verifying the functionality of an input box that takes only integers, etc.

Although there are many methods for assertions, the generic syntax is:

1)- Assert.Method(actual, expected)

2)- Assert.Method(actual, expected, message) ←Message: A string message to display only in case of an error when the assert fails. For example if assert is failing Java exception message is shown, but using that sintex will display the massage mentioned in the parameters instead of Java exception message.

Бывают

**Hard assertions** - stops execution after a failure and moves to the next annotation

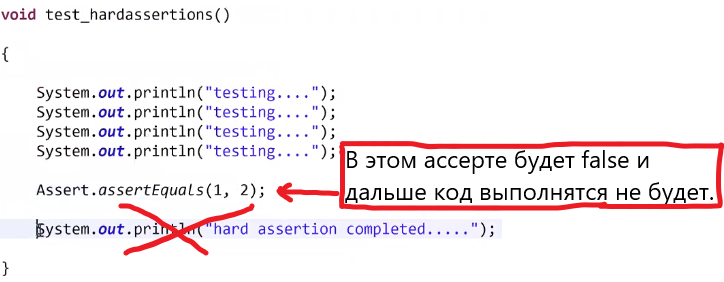
Hard Asserts are those asserts that stop the test execution when an assert statement fails, and the subsequent assert statements are therefore not validated. It plays a vital role in projects where we have an element without whose validation, asserting other elements is useless. One good example in such cases is the login functionality. If I want to see my past orders, for example, then what is the point of checking this test case when the login validation already failed? Hard asserts are the default type of asserts in TestNG,

**Soft assertions -** continue execution after a failure and moves to the next statement line.

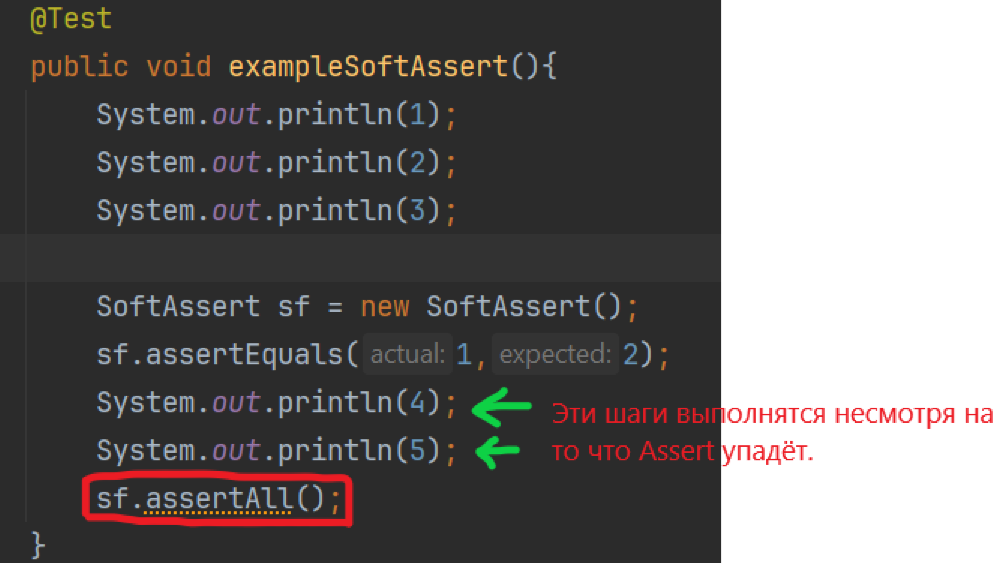
Soft asserts are just the opposite of hard asserts. In soft asserts, the subsequent assertions keep on running even though one assert validation fails, i.e., the test execution does not stop. Soft assert does not include by default in TestNG. For this, you need to include the package org.testng.asserts.Softassert. So, when should we use soft asserts in TestNG? We use soft asserts when we do not care about the failure of specific validations and want the test execution to proceed and also want to see the exception errors.

(Example of hard and soft assertions. В тесте @Test есть два assert-a №1 и №2, которые по сценарию у нас оба должны упасть(не важно какой функциональности) они обычные (т.е. не SoftAssert). Особенность HARD ASSERT что если №1 падает, в консоли выводится (если оно есть) сообщение от данного asserta, то №2 даже не выполняется и вообще если после ассерта №1в тесте есть какие-то инструкции то они также не выполняются, коомпилятор идёт к следующей annotation. ЕСЛИ на этом этапе поменять assert на soft Assert то тест пройдет успешно (оба asserta но почему не понятно), хотя мы знаем что он должен упасть. Всё дело в том что мы не добавили ключевой метод softAssert.assertAll() в конце нашего метода.При использовании softAssert он обязятелен. Добовляем, перезапускаем, оба ассерта зафэйлились и кинули два сообщения от обоих assertov. ВОТ В ЭТОМ И РАЗНИЦА SOFTaSSERT И HARDaSSERTA, при использовании soft тест продолжает выполнятся, тогда как при hard коомпилятор переходит на другой тест. И ещё, если мы используем softAssert и первый assert падает, а второй проходит, то тест всё равно фейлится но в консоли будет сообщение только от того asserta который упал, а так как второй успешно прошол то от него сообщения соответственно не будет). **It is good practice to use hardAssert if you verify only one result, but if you verify more than one result berrer to use softAssert.**

Пример hard assert-a. Hard assert доступны по умолчанию

****

Для Soft assert-ов надо создать экземпляр класса SoftAssert. И этот обьект имеет такие же методы как и HardAssert. Можно делать сколько хочеш ассертов в методе, если один из них упадёт, программа продолжит выполнение. Но в конце надо обязательно вызвать метод .assertAll()



AssertSame - verifies that 2 objects refer to the same object

AssertNotSame - verifies that 2 objects do not refer to the same object

AssertNotNull - verifies that object is not null

AssertEquals - verifies actual and expected results

Assert.assertEqual(String actual, String expected): Pass the actual string value and the expected string value as parameters. Validates if the actual and expected values are the same or not.

Assert.assertEqual(String actual, String expected, String message): Similar to the previous method just that when the assertion fails, the message displays along with the exception thrown.

Assert.assertEquals(boolean actual, boolean expected): Takes two boolean values as input and validates if they are equal or not.

Assert.assertTrue(condition): This method asserts if the condition is true or not. If not, then the exception error is thrown.

Assert.assertTrue(condition, message): Similar to the previous method with an addition of message, which is shown on the console when the assertion fails along with the exception.

Assert.assertFalse(condition): This method asserts if the condition is false or not. If not, then it throws an exception error.

Assert.assertFalse(condition, message): Similar to the previous method but with an addition of a message string which is shown on the console when the assertion fails, i.e., the condition is true\*.