Например у меня есть тест кейс состоящий из одного класса, в классе есть несколько методов, когда тест кейc выполняется, некоторые методы могут pass, некоторые failed или skipped. На основании этих данных я хочу (например) произвести определённые действия:

- если тест failed то я хочу сделать screenShot,

- если pass то хочу сделать то-то,

- если skipped то выполнить это.

Все эти действия (которые будут делатся после failed или pass или skipped) называются **post actions** и чтобы настроить эти post actions надо использовать концепт test listeners. Создать методы в отдельном классе которые (с помощью xml + спец тега будут) задействоваться каждый раз когда метод-test failed или pass или skipped.

Ещё раз! тест листенер - это когда мы хотим выполнить какое то действие после того например как в основном тест кейсе какой то метод-тест зафейлился

Чтобы определить (создать)тест листенерс (класс с методами) есть 2 подхода.

1) **интерфейс ITestListeners** в этом интерфейсе методы. Можно создать свою реализацию (т.е. класс)

2) **класс TestListenerAdapter**, этот класс реализует данный интерфейс и соответственно все методы его. Можно воспользоваться методами или override их в своём классе (отнаследовавшись) под свои нужды.

Некоторые из методов класса TestListenerAdapter:

onTestStart()

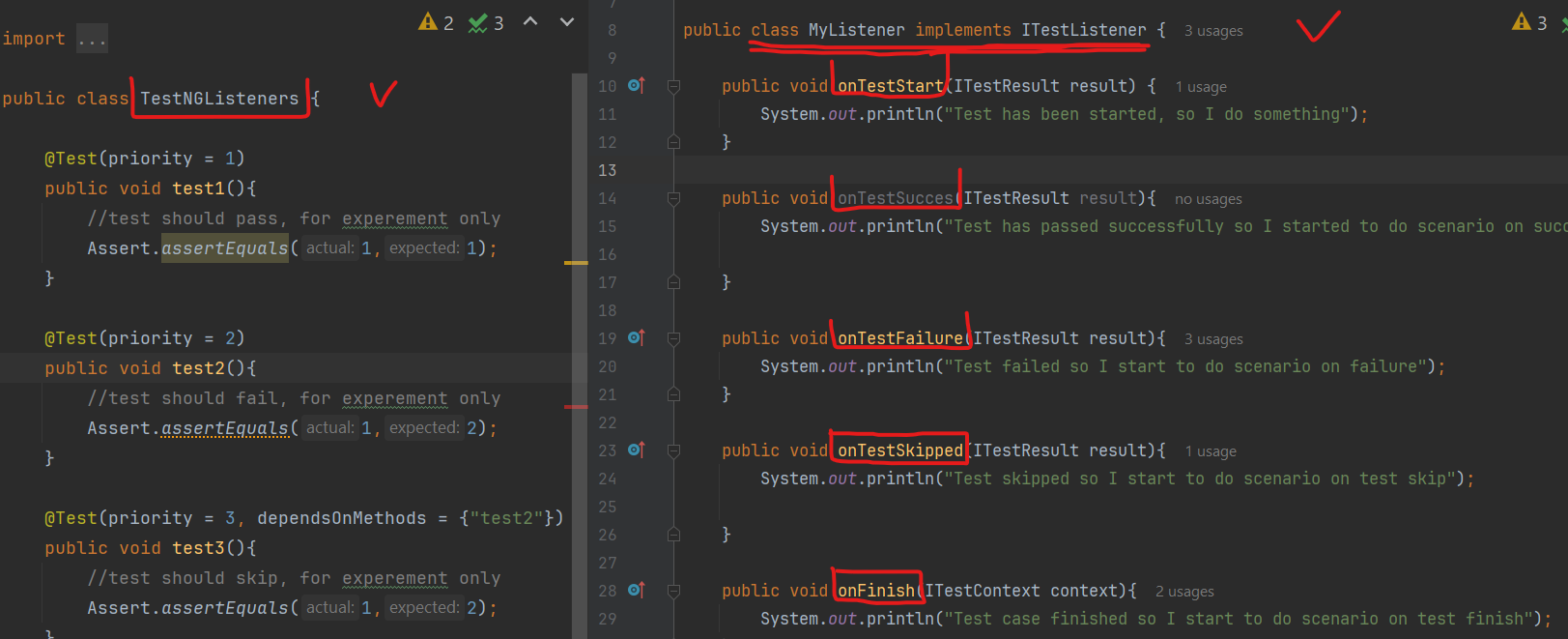
onTestFailure()

onTestSucces()

onTestSkiped()

Внезависимости как будет реализован тестлистенер концепт, методы из него **не** будут вызываться самостоятельно, **они будут тригирится** в зависимости от результата выполнения методов-тестов.

Пример:

Есть класс(test case) слева и класс реализация ITestListeners, справа. 

Test case ( с лева)- это класс с методами заглушками, чисто для примера.

1 метод чтобы просто прошол,

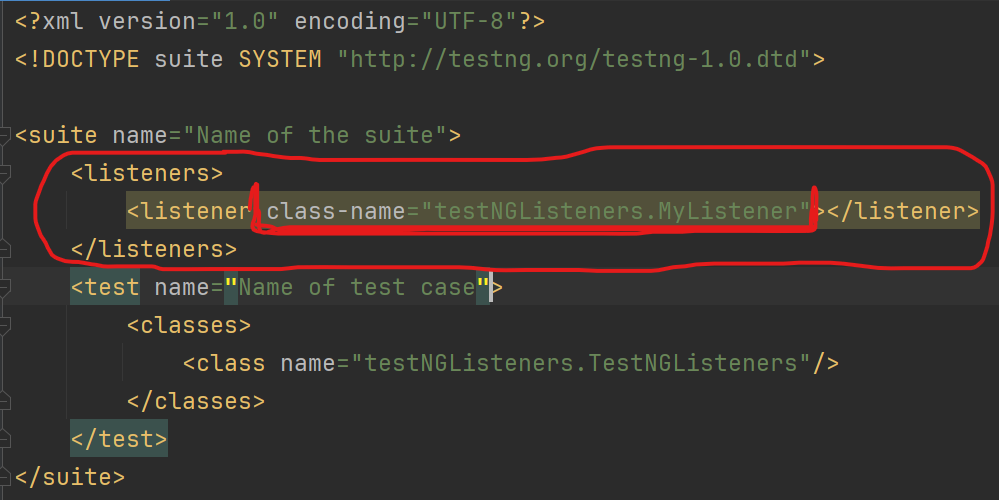
2 - чтобы упал,

3 - чтобы скипнулся

Мой листенерс класс - методы тригерируются на каждый из сценариев из Test case, т.е. когда начнётся выполнятся любой метод-тест (с аннотацией @Test) то это стригерит один метод, когда метод-тест успешно пройдёт то выполнится другой метод, когда метод-тест упадёт выполнится тоже другой метод, когда метод-тест скипнется опять другой метод и в конце по отработке всего Test case-а опять выполнится метод.

Но как мой тест кейс узнает про реализацию класса листененрс? и как они между собой договорятся?

Нужен .xml файл -



а в нём тег <listeners> в нём <listener> и обязательно атрибут class-name = “название Мой листенерс класс”

Ну и в <class> указываем наш тест кейс класс

**ОКАЗЫВАЕТСЯ** можно привязать тест кейс(класс) и листенер класс и без xml. Для этого в тест кейсе (класс отвечающий за тест-методы, **не** за реализацию листенера) над сигнатурой класса указывается

**@Listeners(расположение листенер класса)** аннотация с локейшенам где расположен тест листенер класс.



Но у этого подхода есть минус, если у тебя только один тест кейс(класс) то ок, но если классов несколько то предётся писать эту херню в каждом классе.

ЭТО КАКАЯ-ТО ХЕРНЯ

Listeners are TestNG annotations that literally “listen” to the events in a script and modify TestNG behavior accordingly. These listeners are applied as interfaces in the code. For example, the most common usage of listeners occurs when taking a screenshot of a test that has failed and the reason for its failure. Listeners also help with logging and generating results.

**Benefits of using TestNG Listeners with Selenium**

+ Enhanced Test Reporting: By implementing listeners, you can capture and log events occurring during test execution, such as test case start, test case failure, test case success, etc.

+ Test Result Analysis: Define custom actions to be taken when a test fails, such as capturing a screenshot, logging additional information, or sending a notification. This enables you to take immediate corrective actions and facilitates efficient debugging.

+ Test Data Manipulation: Listeners provide hooks to modify or manipulate test data during runtime. You can use listeners to dynamically update test data, parameters, or configurations before or after each test case execution.

+ Test Execution Control: Listeners allow you to define conditions and logic for executing or skipping tests based on specific criteria.

+ Test Parallelization: TestNG parallel test execution and listeners play a crucial role in parallel test management. This can significantly reduce the test execution time, enabling quicker release cycles.

+ Custom Test Execution Behaviors: The flexibility to define custom behaviors during test execution allows you to tailor the testing framework to suit your requirements and enhance (увеличить) test automation capabilities.