***Тут мало что понятно сразу иди в Обьяснение № 3. Это ниже***

*Note*: *TestNG Parameters are run through the TestNG XML file and not from the test case files directly*.(т.е. воспроизводить надо .xml fail)

-Parametrization starts with: parameter tag and annotation

The purpose of **parameter tag** is to define name and value of the parameter. (it is written in .xml file)

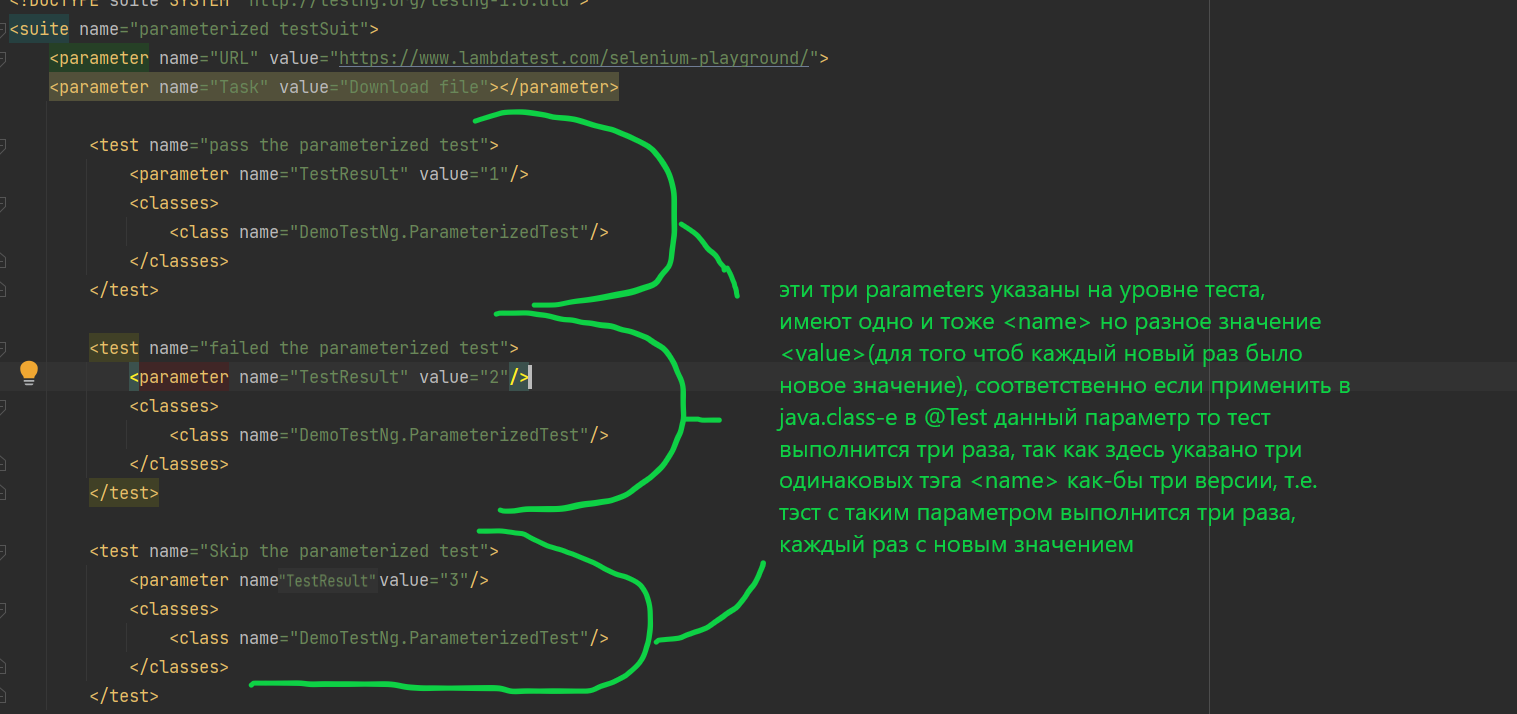
The purpose of **parameters annotation** is to specify how to pass parameters, wich parameter to pass to the method (it is written in java.class file)

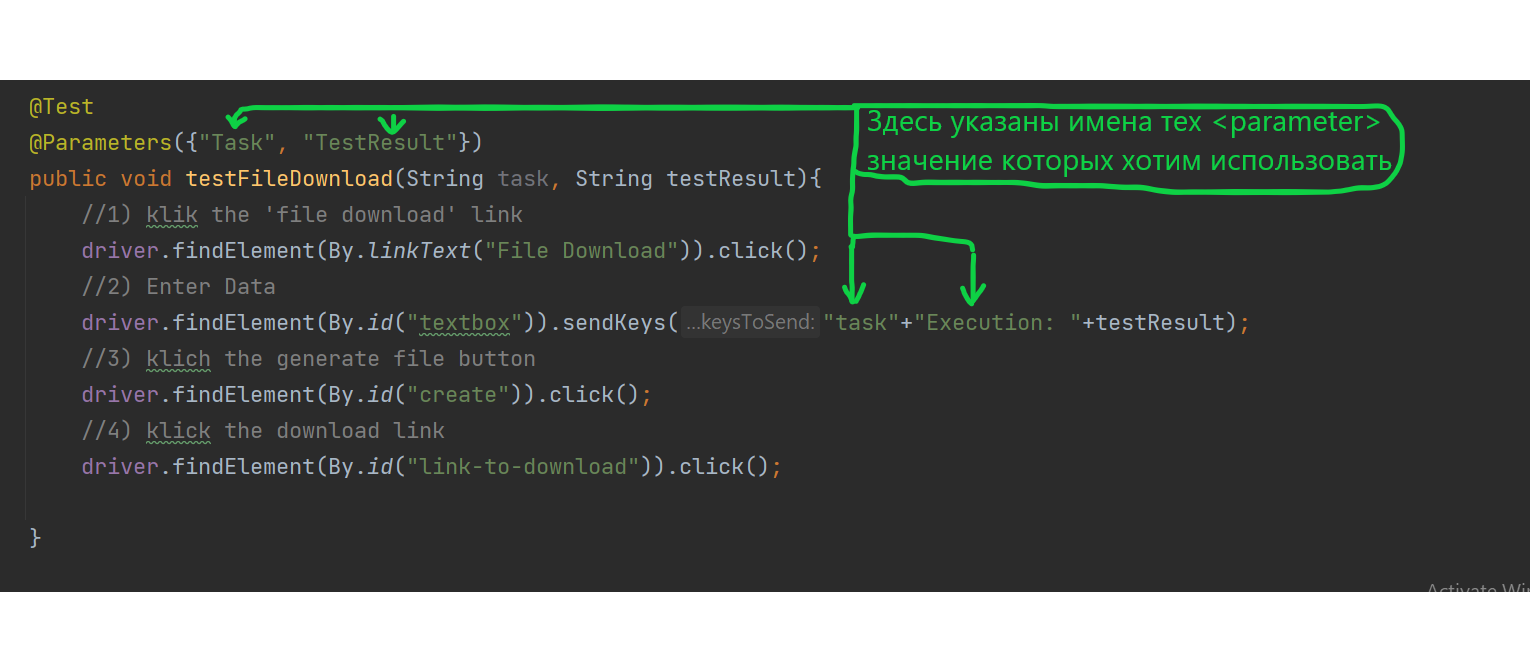
so for project in wich one wont to use Parametrization we need to define .xml file (for parameter tag), and

java.class for parameters annotation

В .xml файле прописываются параметры с помощью тэга <parameter> у этого тега есть атрибуты name и value, в них указываются имя и значение соответственно.

Далее в классе .java когда хочешь применить данный параметр необходимо сверху или с низу от аннотации прописать @parameter в скобках указать название параметра (тот который указывался в атрибуте name), а в скобках метода который помечен аннотацией указать тип значения и переменную, дальше можно использовать данную переменную в теле метода. Ниже скрин где определено три <parameter>

Мы знаем что parameter с name TestResult в .xml файле определён 3 - раза



соответственно метод testFileDownload будет вызван 3 раза с разными значениями. Я тут немного почитал и кое-что передумал. На самом деле тэг parameter может быть определён как в тэге <suit> так и в тэге <test>, тэг <test> определяет параметр, а затем класс в котором будет применятся данный параметр, если всё оставить на этой глубине вложенности (на уровне класса) то данный параметр применится ко всем методам в классе(ко всем тестам) и в примере выше определено 3 параметра на уровне класса, так как по сюжету у нас в классе только один метод, то все эти параметры по очереди прогонят данный тест. В какойто мере получается, что ход программы формируется в .xml файле, сколько в нём тэстов столько раз он и будет ранить эти определённые тэсты. Если добавить в класс ещо один метод и даже определить для него параметр и вызвать через @parameter, тэст зафэйлится так-как к нему придут и те параметры которые были определены ранее, так как они определены на уровне класса. Выход - определять параметры на уровне методов

<test name="The Name Of test">

<parameter name="forSomeMethod" value="Privet">

<classes>

<class name="DemoTestNg.ParameterizedTest">

<methods>

<include name="someMethod"/> //←здесь указываем метод для которого определяем данный параметр

</methods>

</class>

</classes>

</parameter>

</test>

Т.е. получается для каждого метода свой/свои параметры. Но для этого надо определять параметры на уровне методов

И ещо момент- если параметр определён в suit-e то им можно пользоватся в любом методе в java klass-e, в том числе и комбинировать с параметром/параметрами определёнными только для данного метода, для этого перечисли атрибуты name в @parameters(... , …)

Параметр определённый на уровне <suit> можно переопределять на уровне <test>, для этого используй такое-же имя параметра.

**<suite name="Parameter test Suite" verbose="1">**

**<!-- This parameter will be passed to every test in this suite -->**

**<parameter name="suite-param" value="suite level parameter" />**

**<test name="Parameter Test one">**

**<classes>**

**<class name="com.howtodoinjava.test.ParameterTest">**

**<methods>**

**<include name="prameterTestOne" />**

**</methods>**

**</class>**

**</classes>**

**</test>**

**<test name="Parameter Test two">**

**<!-- This parameter will be passed this test only -->**

**<parameter name="test-two-param" value="Test two parameter" />**

**<classes>**

**<class name="com.howtodoinjava.test.ParameterTest">**

**<methods>**

**<include name="prameterTestTwo" />**

**</methods>**

**</class>**

**</classes>**

**</test>**

**<test name="Parameter Test three">**

**<!-- Overriding suite level parameter -->**

**<parameter name="suite-param" value="overiding suite parameter" />**

**<!-- Test specific parameter -->**

**<parameter name="test-three-param" value="test three parameter" />**

**<classes>**

**<class name="com.howtodoinjava.test.ParameterTest">**

**<methods>**

**<include name="prameterTestThree" />**

**</methods>**

**</class>**

**</classes>**

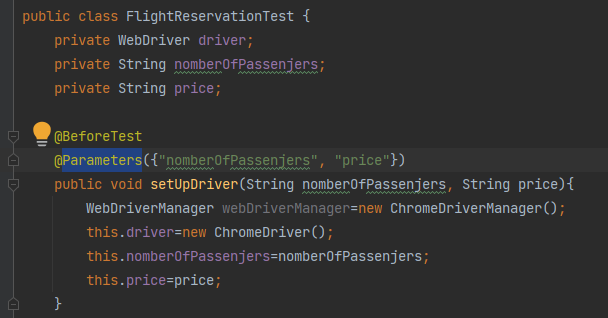
**</test>**

**</suite>**

The values for parameters are always String

One more example

есть class (не важно какой) с полями, и эти поля инициализируются в методе который благодаря аннотации @beforeTest выполняется соответственно перед тестом (первый в классе) в этот же метод с помощью аннотации @Parameters, передаём атрибуты , которые затем перехватывает сам метод (указывает их как тип String) в теле метода значение пораметров присваивается полям класса.

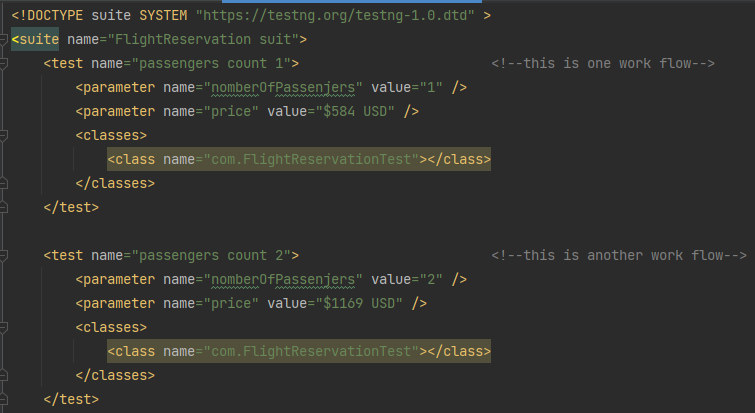


Также чтобы библиотека TestNG понимала откуда брать то параметры необходимо эти параметры предоставить в .xml документе. Синтаксис не сложный, смотри фото

suite <- это большой тест (что то типа сценария, типа выполни одно действие но с разными аргументами)

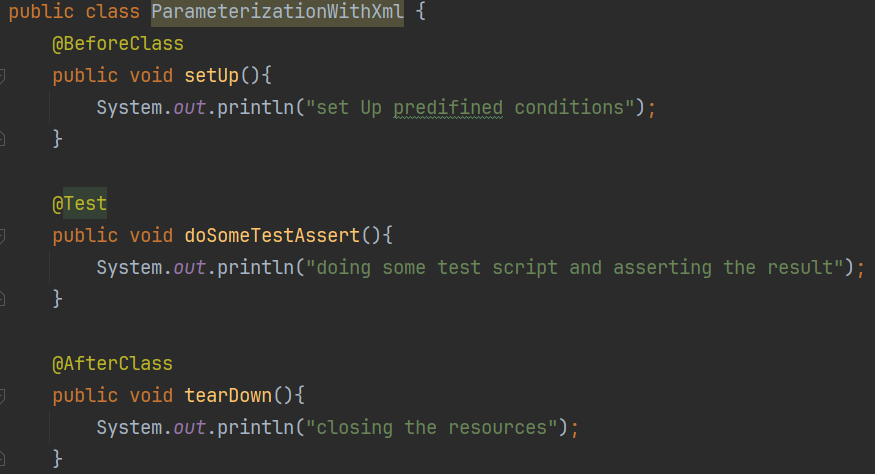
test <- это одна прогонка приложения с одним набором аргументов. Соответственно на фото один и тот же тест выполняется 2 раза но с разным набором аргументов.

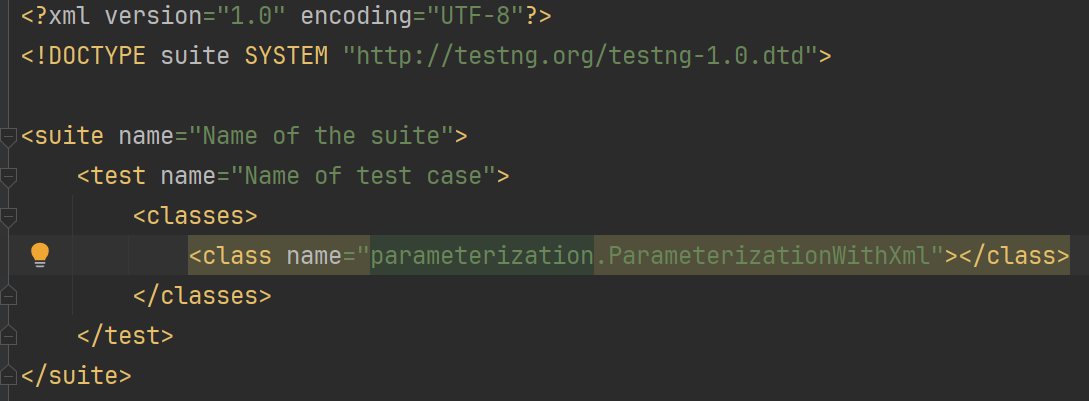
class <- это класс который мы хотим прогнать (runner)



Обьяснение № 3

есть класс



Есть xml который его запускает

Задача: передать параметр (просто какую то String в setUp метод) с помощью XML файла.

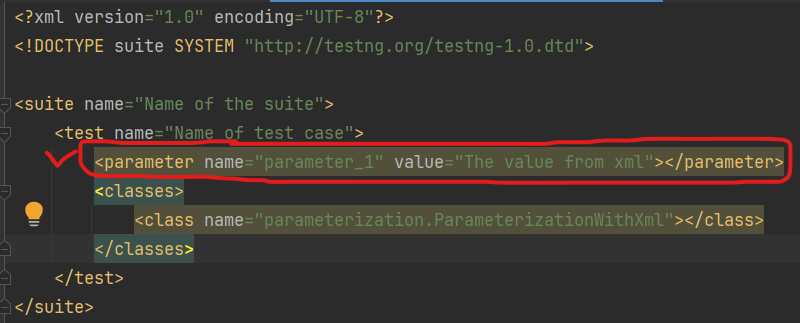
Для этого:

У этого тега есть обязательный атрибут name и value Их заполнять обязательно.

**Name**, с помощью его мы указываем уникальное имя параметру

**Value** = это собственно значение которое хочеш передать в метод.

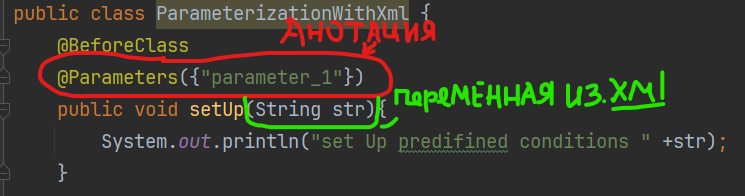
Теперь наш xml выглядит вот так



2)-В классе, непосредственно в методе, для которого предназначен параметр добавляем аннотацию

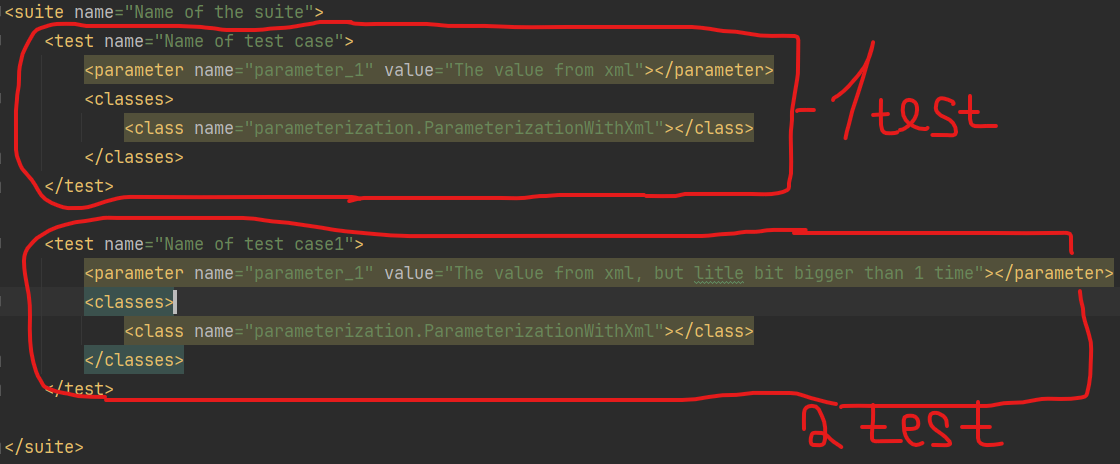
**@Parameters({name\_параметра})**, name\_параметра - это имя параметра которое присвоили в xml файле,и **в параметрах/аргументах** этого же метода принимаем переменную которая и является переменной из xml.

Теперь наш метод выглядит вот так

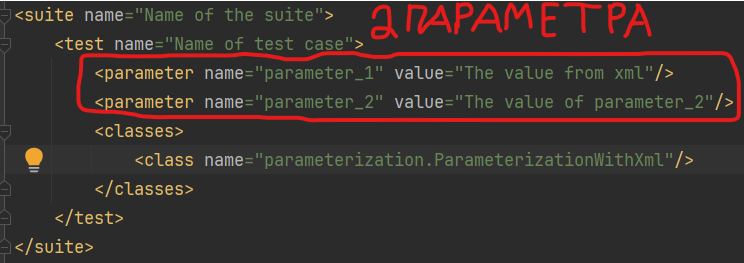


Получив переменную в метод мы можем обрабатывать её как хотим, с любой логикой.

**Теперь!** Например я хочу выполнить свой тест 2 раза **ПО ОЧЕРЕДИ** с разными параметрами. Для этого надо изменить xml файл, и добавить ещё один <тест> тег. В результате класс с методам выполнится 2 раза, столько сколько <test>-ов в XML файле.



**ТЕПЕРЬ!** Например я хочу передать в метод несколько параметров, для этого в xml надо создать несколько <parameter> для <classes>.



А в методе:

1) добавить ещё одно название параметра

2)добавить ещё одну переменную

