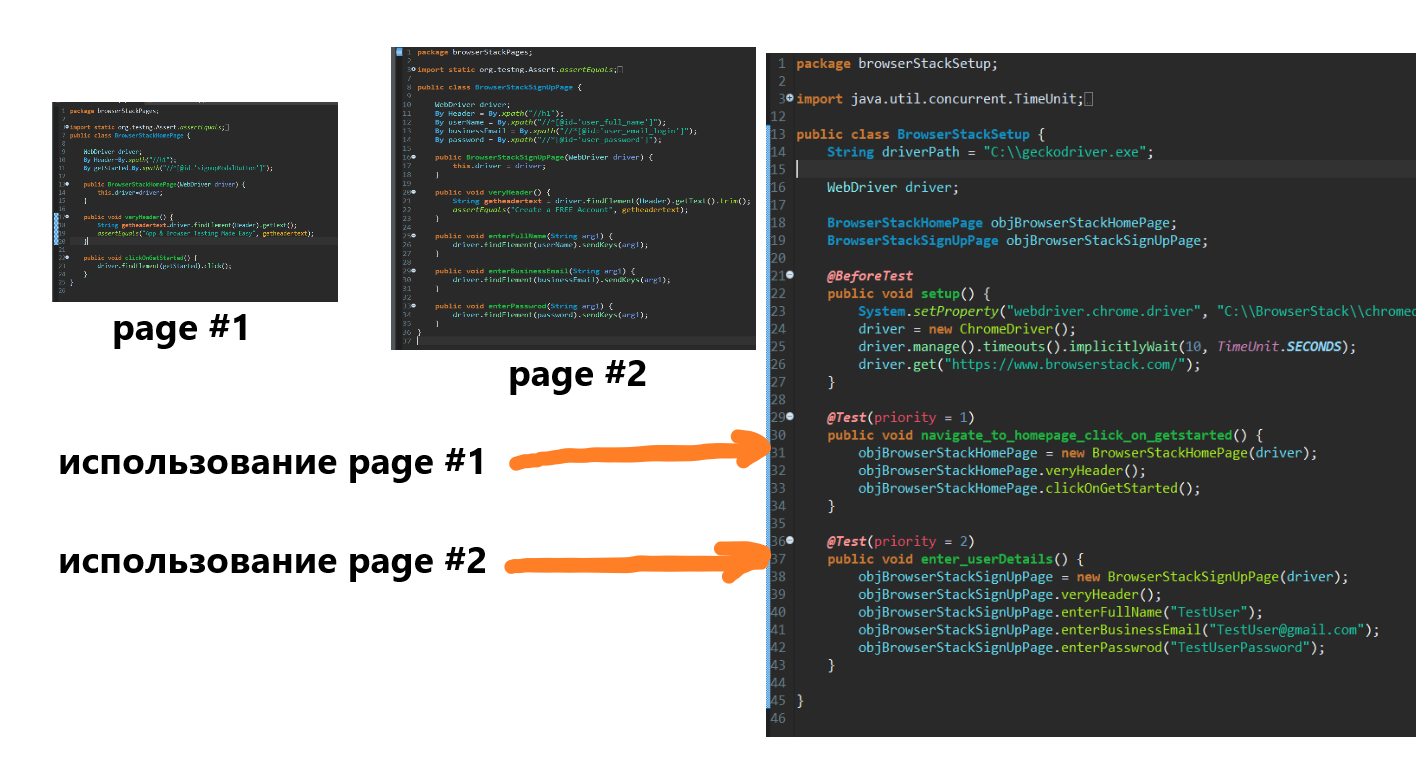
Page Object Model, consider each web page of an application as a java.class file. Each class file will contain only corresponding **web page** elements. Using these elements, testers can perform operations on the website under test. Т.е. каждой веб странице соответствует свой Джава класс в котором есть обьекты (соответствующие веб странице) и методы которые эти обьекты совершают. Всёооо!

Advantages of Page Object Model

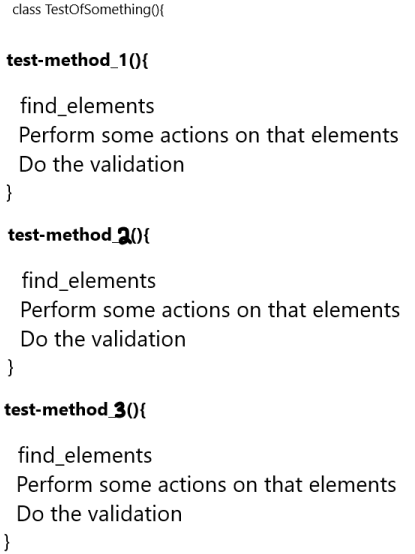
**Easy Maintenance:** POM is useful when there is a change in a UI element or a change in action. An example would be: a drop-down menu is changed to a radio button. In this case, POM helps to identify the page or screen to be modified. As every screen will have different Java files, this identification is necessary to make changes in the right files. This makes test cases easy to maintain and reduces errors.

**Code Reusability:** единажды определив класс с веб элементами и методами, мы можем использовать его в разных тест-кейсах.

**Readability and Reliability of Scripts:** When all screens have independent java files, one can quickly identify actions performed on a particular screen by navigating through the java file. If a change must be made to a specific code section, it can be efficiently done without affecting other files.



Если посмотреть функционально на каждый тест метод в тест кейсе то увидим, что шаги в некоторых их них очень схожи. Сначала мы ищем в дом элементы, затем что то делаем с ними (кликаем, считываем данные, копируем…), затем делаем асерт.



Так может случится,что элементы найденные в метод-тесте 1 нам понадобятся в метод тесте 2, в этом случае придётся перенаходить их используя в принципе один и тот же код, т.е. определять переменные, использовать те же локаторы, всё это будет одинаковое. А потом может оказаться, что это же нам надо и в тест-методе 3…, т.е пишем, фактически один и тот же код.

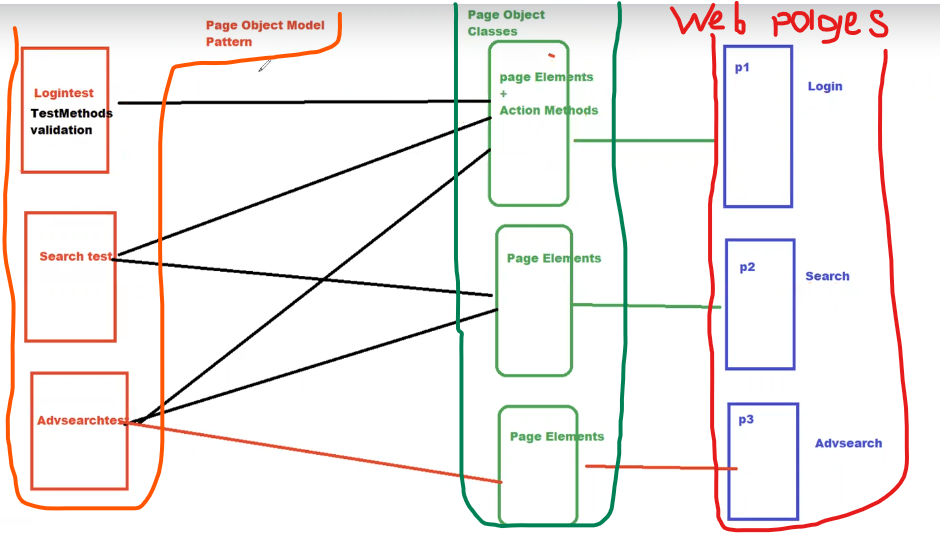
С другой стороны если что то поменялось в локаторе элемента, то так как мы используем этот элемент во многих тест-методах надо идти и везде его исправлять.

Если мы собираемся использовать POM model:

1)- надо сначала определится сколько web страниц имеет наше приложение.

2)- Для каждой web страницы нужно создать отдельный .class в java - it is called Page object class. В каждом из них будут элементы (только одной страницы) и методы над этими элементами.

3)- Для того чтобы писать тест-методы мы используем соответствующие (уже нами написанные) классы. Это позволяет использовать один Page object class в нескольких (сколько надо) тест-методах.



**вариант\_1** каждый Page Object Class - может состоять из:

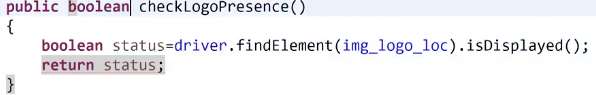
-**конструктора** - чтобы **получить** драйвер при создании экземпляра Page Object Class-а, инициализировать свой драйвер в этом классе нельзя, он будет создан в тест кейсе - классе и передан сюда в Page Object Class.

-**локаторов** (с помощью класса by)

-**методов** в которых делаем что то над элементами расположенными по локаторам

Также важно не делать никакого рода проверок (ассертов) в методах этого класса. Как вариант вернуть булеан а в тесте-методе уже сделать ассерт.



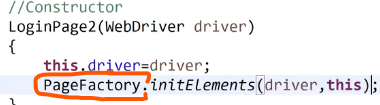
Получаем какой то прообраз utility класса который можно будет использовать много раз в разных тест-методах.(и наверно тест кейсах)

Чтобы использовать Page Object Class, в классе (тест-кейсе) надо создать объект нашего Page Object Class-а но сделать его экземпляром класса у тест кейса (класса), а не какого то отдельного его метода-теста, это позволит использовать этот объект в разных тест-методах.

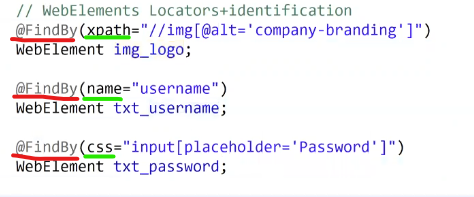
**вариант\_2**

Page Object Class можно составлять с помощью Pagefactory, вместо локаторов мы будем сразу определять webElements. Но делать это не с помощью findElement() метод, как мы делали это раньше(driver.findBy…) а использовать аннотацию **@findBy.** Она автоматически находит элемент (без класса by) и ложит в переменную.

Для этого надо видоизменить конструктор добавив в него инициализацию PageFactory, для этого ему надо передать драйвер и this - обозначающее реализацию в этом классе. Это всё делаем в конструкторе (получаем драйвер + инициализация Page Factory через его метод.)



После этого нахождение элемента и запись его в переменную типа WebElement, синтаксис выглядит так



искать как ты понял можно с помощью разной стратегии (xpath, name, css).

Если надо найти несколько элементов, синтаксис такой же но записывать их надо в соответствующего типа переменную, здесь также хорошо работает list

