**Static keyword in step definition**

Most of the time we use static keyword for not duplicating the initialization of the driver in every scenario. So for example we create a static field

public static ChromeDriver driver

When we run all package the first scenario initiates it so we can use the value of it in the next scenarios and steps and not create it again. But:

1)-we lose ability to run various scenarios independently

2)- when we will have multiple .feature fails we may encounter the error because cucumber may not execute scenarios in a way we expect.

3)-Parallel execution. We got only one instance of driver. When the first scenario execute it will set the driver, and since it is static it will hold the value. But at the same time the second scenario will also execute during parallel execution. So it is also going to use the same driver instance (driver instance - that is the same browser, but we are already use it). So it also going to perform actions on the same browser. Basically 2 different scenarios are trying to perform UI actions on the same browser. So it might create conflict and one of the scenarios might fail. Probably both the scenarios will fail.

**The takeaway** is do not use static keyword since it is going to create a lot of problems

**duplicate UI element definitions**

For example in one step definition you find webElement and perform actions on it and then in another step definition you again find the same webElement and perform actions on it. This approach is wrong. In this case a good practice will be to create an instance of the class and not the step, and you will get rid of some duplicate code. But this elements will be available only in this particular step definition class. In case you got another step definition class where you gonna use the same elements then you are gonna defined them again. S we need to avoid this kind of duplication as well. And this is where Page Object Design comes in picture. Since we are gonna use the same WebElement only on the same page!!! So it is a good practice to keep one step definitions class attached to one domain (web page).

**synchronization strategy**

Thread.sleep(5000); <- - it is very bad practice and it will cost to errors and delays (sometimes the internet is slow and it might take more time to accomplish the step), it is not reliable. Bad practice. And meantime we can-not wait too long since we may hide defects by doing so. Usually for API call timeout is 30 sec.

We need to define some acceptable waiting time in our automation framework.

**Lack of domain object**

In cucumber expression when we use default parameters it is bad practice, it should be domain objects.

****

One more example:



The method above use a lot of information, we could pass 1 or 2 objects which contains all information in the fields.

The answer is: use custom parameters.

**dependent scenarios**

Many people write their scenarios in a way that they are dependent on each other.

1)-They may be sharing the same instance of driver - in case of parallel execution it may not work. Creating a separate driver instance for each test case / scenario is a important selenium guideline.

2)-Reuse the same test data

3)-The same user

4)-User state. Some of the scenarios may expect clean user state, so you will not be able to run it in parallel execution

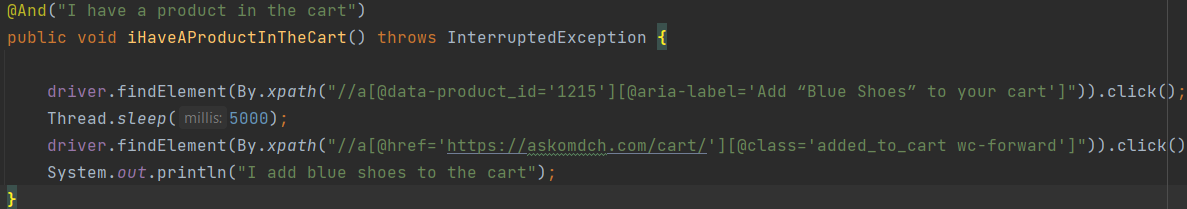
5)-Application state - each scenario should start from a clean application state. It should not start from the state that was left by the previous scenario. If the previous scenario fails and does not reach the point that was expected by the second scenario to start then both scenarios will fail. And the second scenario will give you a negative outcome only because it did not meet the expected conditions to become. So each scenario should start from a clean state and exit application state.

Cucumber designed in a way that you write an independent scenario.

**Using UI for creating Application statement**

It is highly not recommended to work with UI, since it might be easily changed (and unstable due to the internet speed). If our task (step) is not straight forward depend on UI, use API. Because the intention of this test is not to test the text on the add to cart button. The intention is to test that the customer is able to place the order. So it should not fail if the text on the button changes.

For example:



For that product we might not work with UI at all. We can work with Data Base or API, because we don't care how the product is added to the cart, by the scenario it should be added that we can proceed to the next step in the scenario.

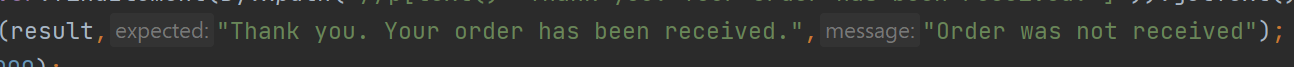
**Hardcoding**

Remove hardcoding as much as possible

<- - this one for example needs to be replaced.

In this particular case we should place the URL in some configuration file and end point in some class or in some file like JSON

One more example of hardcoding:



This text we can put in an XML file and read it from there and not keep it in the step definition. In that way it is easy to manage / maintain this text. Because there is possibility that we may want to use it in a different assert.

**Incompatibility error**

It is when for example we have hardcoded manually initialization of driver



In this case when the browser will get updated we will have incompatibility between driver and browser . In the local machine it is ok no problem, but in the cloud, when the automation runs in the CI environment this may not be desired. So we need to do it automatically. There is a library that we can use “Webdriver Manager”, it will manage the driver automatically.

**missing the multi browser support**

It should be done automatically with the aid of Maven.

**support for different environment**

It is very possible that in a project you may have different environment like: QA environment. stage environment, production environment. So for each of this environment the base URL will be different. In that case the automation should be designed such that that it should run on any of this environments without requiring any change in the code. So I should be able to take base URL as a the parameter in the Maven command and I should be able to execute the automation for a specific environment.