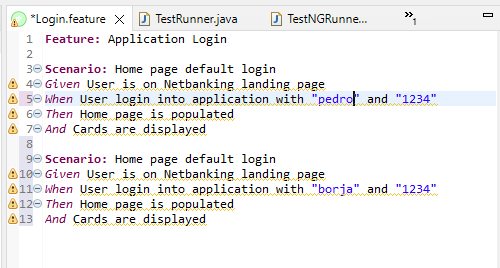
Seccion 4: Code Reusability with Regular Expressions

Punto 1: Usar expresiones regulares.

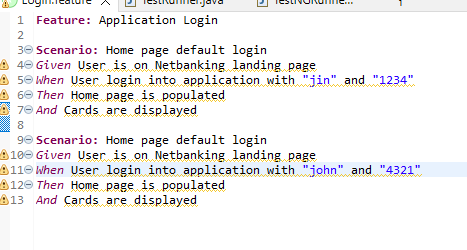
Vamos a aprender a definir las expresiones regulares para poder definir bien los pasos en Cucumber.

No vamos a definir un test case que sea igual. Toda la lógica se definirá en en backend, tantos los casos positivos como negativos. Son los escenarios los que controlarán cada uno de los datos que le pasamos, pero toda la lógica vendrá en el backend.

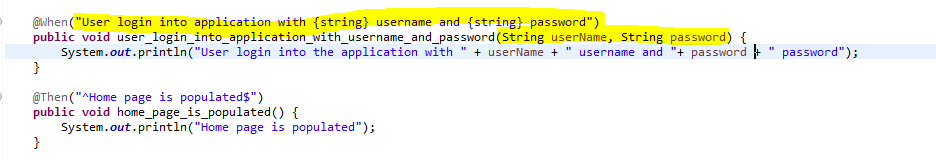


Para hacer esto, vamos a utilizar **regular expressions** para poder definir cada uno de los pasos en la clase con los **Step Definitions**.

Para este caso, lo vamos a escribir como:



Finalmente, en el código de las StepDefinitions

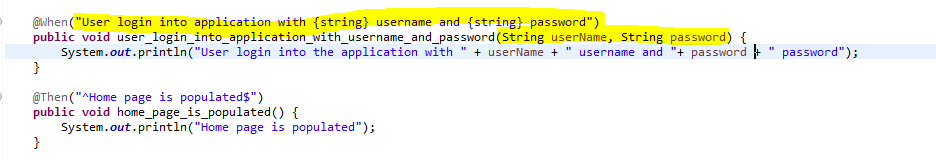


Se ha de pasar los parámetros como argumento de la función

Punto 2: Como reutilizar funciones para diferentes parámetros

Vamos a ver cómo pasar más de un parámetro a la función, parametrizando con Regular Expressions.

Tener en cuenta que se ha de pasar como argumento de la función estos parámetros.



**SECCION 5: Data Driven with Cucumber**

Punto 1: Importancia de la dataTable por los DataDriven tests

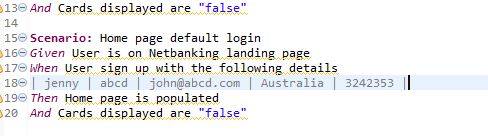
Una de las maneras de pasar datos y correr varios tests, es pasar una tabla de datos en el *.feature****.***

Para ello utilizamos la palabra reservada **Examples.** Además, usar Cucumber nos ayuda a hacer esto de forma sencilla,, ya que no tenemos que usar HashMap u otras formas de pasar los datos.

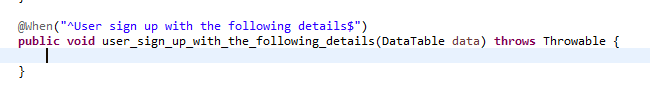
Se los pasamos como una tabla, y tirando.

Vamos a crear una **tabla de datos.**

Para pasar una tabla de datos, lo primero es definirla en el archivo *.feature*



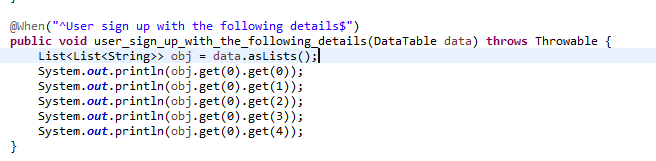
Una vez definida, en la clase con los **Step Definitions,** se inyectan estos datos metiendo como parámetro la clase **DataTable**.



Se le pasa una lista de parámetros y se extraen los valores.

En este caso, hay que tener en cuenta que una tabla se compone de filas y columnas, y por lo tanto, hay que recoger los valores como una lista de una lista de strings.

Luego iteramos



Con esto nos quitamos la necesidad de tener que utilizar un Excel, JSON,... se hace todo en el .feature.

**Punto 2: Parametrizar los test cases con Example keyword**

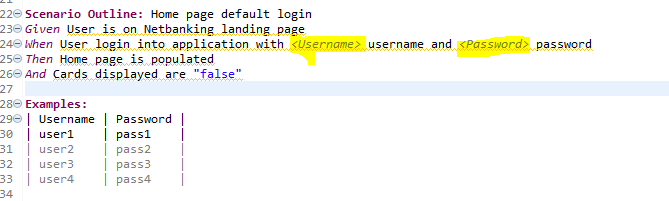
Lo que es parametrizar es repetir los test cases con diferentes data sets.

Para ello, tenemos que utilizar la palabra clave **Example**.

Además, no se utilizan **Scenario**, sino **Scenario Outline**.

De esta forma, por cada fila de datos que se añade, se ejecutará una vez más ese test case (una por cada datos que metamos).

En el archivo de *.feature*, quedaría así:



A continuación, debemos retocar los Step Definitions.

SECCIÓN 6: Cucumber Core

Paso 1: Tagear features para controlar test cases

Uno de los pasos fundamentals a la hora de correr los test cases es que hay algunos que no los queremos correr como el resto de los test cases.

Por ejemplo, pensar que queremos lanzar una regression y solo unos tests tienen que correr, otros no,...

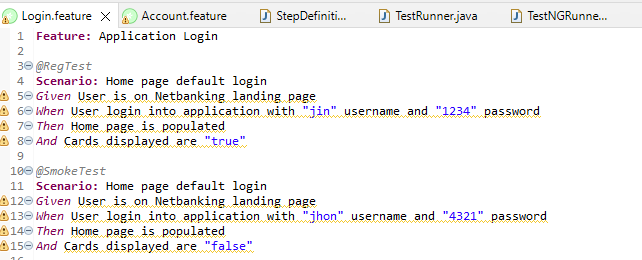
Al igual que hacíamos con TestNG en los testng.xml files, vamos a utilizar **tags para indicar que tests corren y cuales no.**

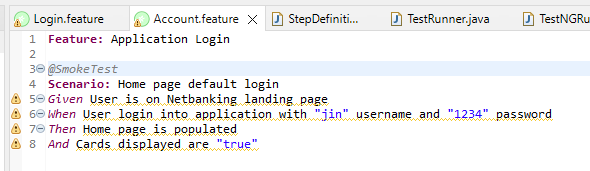
Hay que controlarlo.

Vamos a utilizar la anotación *@RegTest* y *@SmokeTest* para poder definir los casos de prueba.

Creamos otra *.feature* y lo llamamos Accounts

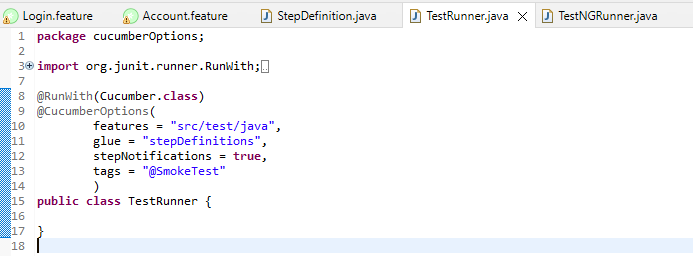
Por otro lado, añadimos las tags al archivo **Login.feature**





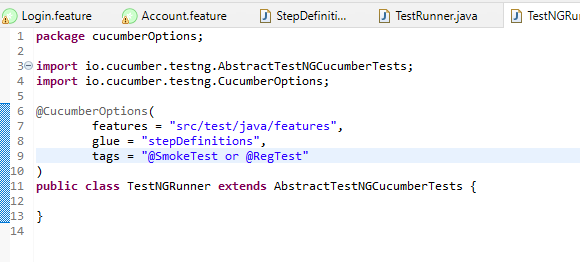
Lo siguiente es definir en el Runner que test cases vamos a correr y cuales no.

Se utiliza el parámetro **tags** para poder definir cuales queremos correr.

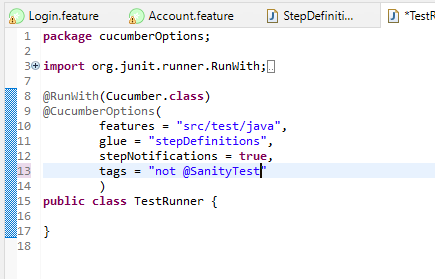
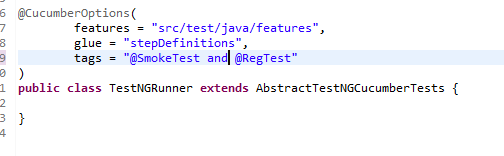


También se pueden correr varios tests a la vez con anotaciones.

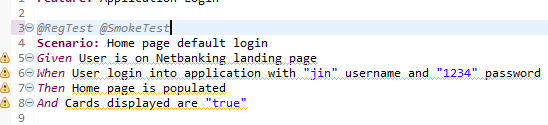
Se utilizan booleanos (con **OR**)



Para quitar unos de los tests



También se pueden añadir múltiples tags a un test case



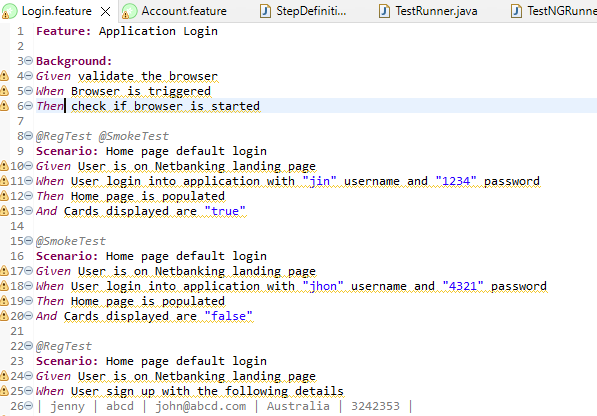
De esta forma, si queremos correr sólo los que cumplan esta condición, debemos utilizar los AND.

**Punto 2: Usar la palabra reservada Background**

Se usa la palabra Background para todo aquello que sean inicializaciones, checkings, ver que está el token.

Es necesario hacer esto como un prerequisito.

Se separa esto del resto de los pasos.



Por último, se han de implementar en el Step Definition, vomo hicimos antes.

De esta forma, separamos el código común a todos los test cases con el código que es el que testea la aplicación.

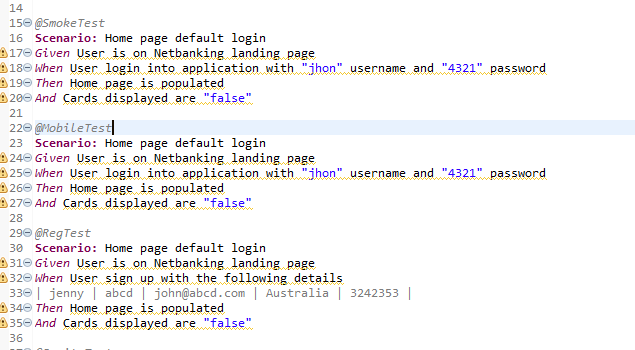
En el caso de tener que definir varios prerequisitos en diferentes test scenarios, no se puede utilizar la palabra reservada **Bacakground**. Para ese caso, se van a utilizar lo que denominamos **Hooks**.

**SECCIÓN 7: SETUP AND TEARDOWN FOR CUCUMBER TESTS**

**Punto 1: Uso de los Hooks para poder gestionar los test cases**

Ahora lo que vamos a hacer es querer separar los tesrts para por ejemplo testear mobile.

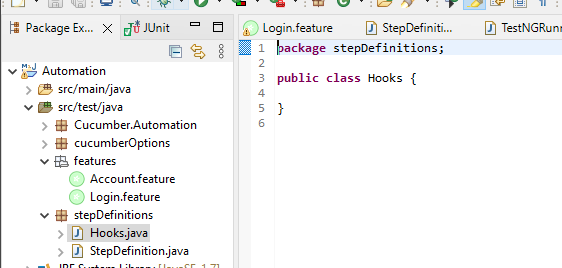
Por ejemplo



En este caso, se repiten los test cases y se pone el tag @MobileTest.

Como no van a utilizar los mismos prerrequisitos que los definidos en el **Background**, vamos a definir los Hooks.

Se crea una clase para ello.



En esta clase se definirán los prerrequisitos que se correrán antes, así como lo que se debe cumplir al final (los teardown)