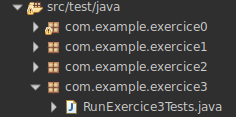
# Exercices BDD/TDD :

* Dans les exercices suivants, on va utiliser les assertions AssertJ au lieu des Assert JUnit.
  + Pour simplifier l’usage d’AssertJ dans Eclipse :
  + Allez dans  Window ▸ Preferences ▸ Java ▸ Editor ▸ Content assist ▸ Favorites ▸ New Type, entrer org.assertj.core.api.Assertions and confirm.
  + Vous devez voir la ligne suivante ‘org.assertj.core.api.Assertions.\*’ dans la liste.
* Commencez par importer le projet ‘BDDExercices’ dans Eclipse

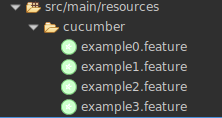
## BDD :

Le projet BDDExercices contient 4 exercices :



* Exercice0 : Basic example
* Exercice1 : With Data-table
* Exercice2 : With Data-Table and multiple results
* Exercice3 : REST Api

Les features sont dans le dossier ‘resources’ :



### Exercice 0 :

* Feature :

**Feature:** belly

**Scenario:** eat cukes

*Given* I have 15 cukes

*When* I eat 5 cukes

*Then* I would have 10 cukes

1 - Lancer le test ‘RunExercice0Tests’.

2 – Dans la console, ***Cucumber vous génère les Steps***.

3 – Copiez les steps dans une nouvelle classe ‘Exercice0Steps’

4 – Relancer le test : ***Tous les tests sont ‘skipped’*** car on n’a pas de code pour le moment. Cucumber nous demande d’implémenter les fonctions ‘Steps’( ***TODO Implement me***).

5 – On va commencer par coder le premier Step :

**private** **int** cukes;

***@Given***("^I have (\\d+) cukes$")

**public** **void** **i\_have\_cukes(int** arg1) **throws** **Exception {**

cukes = arg1;

**}**

6 - on relance les tests. On remarque que le premier step passe.

7 - On code le 2éme Step :

***@When***("^I eat (\\d+) cukes$")

**public** **void** **i\_eat\_cukes(int** arg1) **throws** **Exception {**

cukes - = arg1;

**}**

8 - on relance les tests. On remarque que le 2éme Step passe aussi.

9 – finalement, on va rajouter une assertion dans le dernier Step :

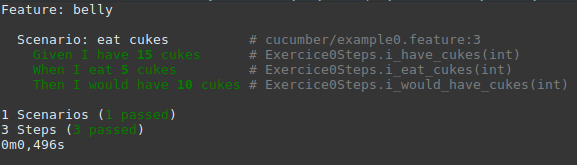
***@Then***("^I would have (\\d+) cukes$")

**public** **void** **i\_would\_have\_cukes**(**int** arg1) **throws** **Exception {**

*assertThat*(cukes).isEqualTo(arg1);

**}**

10 - on relance les tests. Tous les tests passent :



### Exercice 1 :

**Feature:** Shopping

**Scenario:** Give correct change

*Given* the following groceries:

| name | price |

| milk | 9 |

| bread | 7 |

| soap | 5 |

*When* I pay 25

*Then* my change should be 4

1. De la même manière que le 1ér exercice, on lance la classe de test **RunExercice1Tests**
2. On copie les Steps générés dans la console dans une classe **Exercice1Steps.**
3. On lance les tests, les tests sont skipped. Cucumber demande d’implémenter le 1er Step :



1. On commence par coder le 1er Step. On va d’abord créer un Objet Grocery qui contient 2 champ name et price.

**public** **class** **Grocery** {

**private** **String** name;

**private** **int** price;

**public** **Grocery**(**String** name, **int** price) {

**super**();

**this**.name = name;

**this**.price = price;

}

**public** **String** **getName**() {

**return** name;

}

**public** **void** **setName**(**String** name) {

**this**.name = name;

}

**public** **int** **getPrice**() {

**return** price;

}

**public** **void** **setPrice**(**int** price) {

**this**.price = price;

}

}

Puis, on va créer un classe Calculator qui va nous calculer le reste :

**public** **class** **Calculator** {

**private** **static** List<Grocery> *groceries* = **new** ArrayList<Grocery>();

**private** **int** payedMoney;

**public** **static** **void** **push**(**Grocery** e) {

*groceries*.add(e);

}

**public** **int** **getPayedMoney**() {

**return** payedMoney;

}

**public** **void** **setPayedMoney**(**int** payedMoney) {

**this**.payedMoney = payedMoney;

}

**public** **int** **getSum**(){

**int** **sum** = 0;

**for** (**Grocery** **grocery** : *groceries*) {

sum += grocery.getPrice();

}

**return** sum;

}

**public** **int** **getChange**(){

**int** **sum** = getSum();

**return** payedMoney - sum;

}

}

5 - Nous avons choisi List pour représenter ‘Groceries’.On rajoute les données dans la liste :

**private** **Calculator** calc = **new** Calculator() ;

***@Given***("^the following groceries:$")

**public** **void** **the\_following\_groceries(** List<Grocery> list) **throws** **Exception** {

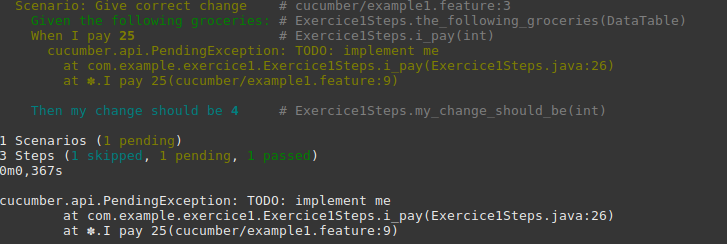
**for(Grocery** **grocery :** list) {

**Calculator**.*push(*grocery) ;

}}

}

6 – On relance les tests, on remarque que le 1er Step passe :



7 – On passe au 2ème Step :

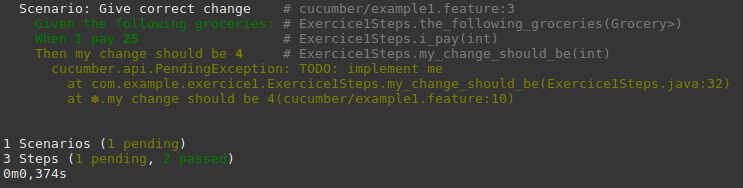
***@When***("^I pay (\\d+)$")

**public** **void** **i\_pay**(**int(**arg1**)throws** **Exception{**

calc.setPayedMoney(arg1);

}

8 – On relance les tests. On remarque que les 2 premiers Steps passent :



9 – On code le dernier Step :

***@Then***("^my change should be (\\d+)$")

**public** **void** **my\_change\_should\_be**(**int** arg1) **throws** **Exception{**{

**int** **change** = calc.getChange();

*assertThat*(change).isEqualTo(arg1) ;

}

10 – On relance les tests et on remarque que tous les tests sont en vert.

### Exercice 2 :

**Feature:** Testing refrences

**Scenario:** Get reference details by id

*Given* the catalog with the following Elements:

| id | name | type | price |

| 1 | BDD book | book | 30 |

| 2 | TDD book | book | 40 |

| 3 | AssertJ website | website | 0 |

*When* I choose the Element <id>

*Then* The name should be <name>

*And* the type should be <type>

*And* the price should be <price>

* On suit les mêmes étapes de l’exercice précèdent.