Test Unitaire

- Un des tests parmi
  - Les tests d'intégration, test utilisateur, test de recette, test de performance, ...
- C'est un test automatisé local à une classe
  - ou un petit nombre de classes qui marchent ensemble
- Sans dépendances, sans se soucier des autres classes
  - pas de base de donnée,
- Conformité (couverture du code)
- Non-Regression

Savoir quoi tester

- On teste les cas qui marchent
  - Renvoie bien les bonnes valeurs
- On teste les cas qui ne marchent pas
  - Valeur indiquant une erreur, exceptions si ...
- On teste les cas limites
  - Cas qui risque de ne pas marcher (expérience !)

Exemple

- On cherche un caractère dans une chaîne
- On teste les cas qui marchent
  - On cherche 'a' dans 'tard"
- On teste les cas qui ne marchent pas
  - On cherche 'b' dans 'tot"
  - On cherche 'c' dans null
- On teste les cas limites
  - On cherche en première et dernière position
  - On cherche avec plusieurs occurrences ... (vérifier le contrat ...)

JUnit 5

- On écrit une classe FooTest si la classe a tester s'appelle Foo

- On utilise l'annotation @Test pour indiquer qu'il s'agit d'une méthode de test

```
public class Foo
{
   @Test
   public void myFirstTest()
   {
        ...
   }
}
```

JUnit 5

- On utilise des méthodes assert* pour vérifier des propriétés

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class Foo
{
   @Test
   public void myFirstTest()
   {
       assertEquals(3, 2 + 1);
   }
}
```

- assertEquals(expected, value) ok si .equals()
- assertSame(expected, value) ok si ==
- assertTrue/assertFalse(value)

Test & Conception

- Pas de test, pas de confiance sur le code
- Pas de test, pas de refactoring
- Pas de test, pas d'évolution du design
- Pas de test, pas de chocolat

3 règles des tests

- Ne pas modifier un test qui échoue
- Les tests n'expliquent pas pourquoi cela plante
- Les tests ne garantissent pas à 100% que cela marche

Ecrire un test unitaire en pratique

- Un test unitaire doit être simple et bourrin
  - Pas d'algo
  - Pas de ruse
- Un test unitaire doit tester une seule et unique chose
- Ne pas regarder le détail de l'implémentation pour écrire un test. Se concentrer sur le contrat

Quand écrire un test en pratique

- Si on a un bug pas couvert par les tests,

on commence pas écrire un test

(qui doit planter !)

  - On reproduit le bug

- Si on veut écrire une API

  - TDD

- Dès que l'on commence à avoir les idées assez claires sur le

design et le code

  - Pas trop tôt, pas trop tard (expérience !)

Test Driven Development

- Pratique de developement qui consiste à écrire

les tests d'abord

  - Garantit que les tests sont exhaustifs
  - On s'oblige pas penser à tous les cas de test
  - Se concentrer sur le contrat et pas sur le comment
  - Le danger, on ne s'occupe pas du comment
- TDD is Dead

http://martinfowler.com/articles/is-tdd-dead/