La méthodologie autour des tests

TD 1: Les tests unitaires

Exercice 1 : Avenger

1/ Créer une classe de test pour la classe Thanos

2/ Écrire un test pour le constructeur vérifiant que Thanos a bien le nombre de pierre qu'on lui donne et qu'il n'a pas réussi sa mission

- Tester en initialisant thanos avec 3 pierre.
- Tester en initialisant thanos avec 7 pierre.
- Tester en initialisant thanos avec -1 pierre.

3/ Tester la méthode gagnePierre.

- Tester en initialisant thanos avec 0 pierre.
- Tester en initialisant thanos avec 7 pierre.

4/ Tester le retour de la méthode claquementDeDoigts()

- Tester en initialisant thanos avec 0 pierre avec un nbPopulation à 7 700 000.
- Tester en initialisant thanos avec 6 pierre avec un nbPopulation à 7 700 000.

5/Tester la méthode toString

- Tester en initialisant thanos avec 0 pierre.

6/ Initialiser deux thanos ayant 0 pierre et testons l'égalité des deux.

- Sans modifier le code de la class. Que remarque t'on?
- Faite généré par l'IDE la méthode equals() et relancer le test. Que remarque t'on?

Exercice 2: Harry Potter

En utilisant la librairie AssertJ

1/ Tester qu'à l'initialisation, il y a bien 4 maisons mais qu'il n'y a pas d'élèves.

2/ Tester la fonction inscriptionEleve("Lucius Malefoy", Serpentard)

3/ Tester la fonction arriveDesHeros()

4/ Tester la méthode findMaison() suite à arriveDesHeros(). Utiliser les annotations @ParameterizedTest et @ValueSource pour faire afficher le message "Test de la maison de 'Harry Potter'"

Exercice 3: Les mondes de Ralph

Un bug s'est glissé dans le code du jeu Vaneloppe. Je l'ai remarqué grâce au test unitaire du développeur. Maintenant il faut le corriger.

- Trouver la ligne qui ne fonctionne pas avec l'aide du débogueur.
- Comment aurait-on pu faire pour que le débogage est été plus rapide?