TD3 - Testabilité – Injection de Dépendances *(Jean-Marie Mottu)*

Nous travaillons sur le cas d'étude d'une classe Chien que vous avez peut-être déjà aperçue en POO mais qui a été redéveloppée pour ce TD. Vous allez progressivement vous rendre compte de problèmes de testabilité, du besoin d'avoir une gestion des dépendances pour permettre de tester.

De nouveau, on considère une approche fonctionnelle en n'exploitant que la spécification pour concevoir les tests. Ici considérons toutes les informations ci-dessous :

- la première ligne de la javadoc spécifie les « exigences »
- les @ de la javadoc complété par la signature des méthodes spécifient le domaine d'entrée et de sortie.

```
1   package but1.iut.r203.chenil
2
3   import ...
6
        mottu-jm *
7   class Chien (nomParam : String, raceParam : String){
8       val nom = nomParam
9       val race = raceParam
            mottu-jm
10      private var dateNaissance : LocalDate? = null
11          set(value) {...}
14
15      /**
16       * Affecte une date de naissance au chien
17       * @param anneeNaissance: Int
18       * @param moisNaissance: Int
19       * @param jourNaissance: Int
20       * @throws  DateTimeException        : quand les paramètres sont incorrects
21       */
            mottu-jm
22      fun setDateNaissance(anneeNaissance: Int, moisNaissance: Int, jourNaissance: Int) {...}
25
26      /**
27       * Calcule l'age en mois du chien
28       * @return age : Long
29       */
            mottu-jm *
30      fun ageMois(): Long {...}
37
38      /**
39       * Calcule l'age en mois du chien par rapport à une date choisie en console
40       * @return age : Long
41       */
            new *
42      fun ageMoisDateConsole(): Long {...}
56
            new *
57      override fun toString(): String {...}
60  }
```

Partie 1 – Premier problème de testabilité

Exercice 3.1. Test fonctionnel de *setDateNaissance()* *(40 min)*

Question 3.1.1. Concevez les tests par approche fonctionnelle de `setDateNaissance()` avec une analyse partitionnelle.

Pour cela reprenez la méthode vue en CM et au TD précédent, en vous limitant à notre siècle.

Remarque :
Est-ce pertinent en 2024 de ne pas considérer des chiens nés au précédent siècle qui auraient plus de 24 ans ?

Quid pour les humains : https://www.bbc.com/news/articles/c9wz7pvvjypo

Exercice 3.2. Problème de testabilité *(20 min)*

Récupérez le code dans IntelliJ depuis ce projet gitlab :

https://univ-nantes.io/iut.info1.qd1.automatisationtests/butinfo1-qd1-td3

Question 3.2.1. Implémentez les cas de test conçus. Quel problème de testabilité rencontrez-vous ?

Exercice 3.3. Amélioration de la testabilité *(20 min)*

Quand un problème de testabilité est identifié, le testeur demande au développeur s'il peut améliorer définitivement le design du logiciel pour lui permettre de tester efficacement.

Question 3.3.1. Prenez le rôle de développeur et résolvez le problème.

Question 3.3.2. Pourquoi est-ce une tâche de développeur ?

Par exemple : https://www.caradisiac.com/Un-radar-fatal-picard-qui-insulte-les-chauffards-77912.htm

Question 3.3.3. Reprenez le rôle de testeur, pouvez-vous implémenter et lancer les tests désormais ?

Des tests échouent, mais ce n'est pas le but aujourd'hui de corriger le code.

Partie 2 – Second problème de testabilité

Nous allons maintenant tester la méthode `ageMois()`. On se passe de faire une analyse partitionnelle ici (vous pouvez la faire pour vous entrainer hors séance), mais nous allons travailler avec des cas de test fournis pour illustrer les problématiques de testabilité.

Exercice 3.4. Problème de testabilité *(20 min)*

Question 3.4.1. Implémentez ce cas de test :

```
CT_age1 : ( « age depuis le 28/2/24 »,        // intention
            création d'un chien quelconque,   // initialisation
            naissance le (2024, 2, 28),       // Donnée de Test
            2 mois)                           // Oracle
```

Question 3.4.2. Lancez-le, passe-t-il ?

Question 3.4.3. Passera-t-il encore dans un mois ?

Question 3.4.4. Quel est le problème de testabilité ?

Question 3.4.5. Quelle variable manque-t-il dans le CT_age1 pour considérer cela ?

Exercice 3.5. Injection de dépendance de méthode *(20 min)*

Notre objectif est de permettre de calculer l'âge par rapport à une date qu'on choisira dans les tests. Pour cela, il faut ajouter un mécanisme d'injection de dépendance plutôt que d'avoir une dépendance interne à une méthode.

Question 3.5.1. Commentez la ligne qui affecte la `dateJour` avec `LocalDate.now()` puisque ce `now()` n'est pas contrôlable. Quel est le problème maintenant ?

Question 3.5.2. Où faire l'affectation de `dateJour` ?

En cours, nous avons étudié une injection de dépendance au niveau de la classe, mais est-ce que ça a du sens conceptuellement que la date du jour à laquelle on veut calculer son âge soit une caractéristique d'un chien ?

Question 3.5.3. Déplacez la déclaration de `dateJour`.

Question 3.5.4. Implémentez le cas de test amélioré à la Question 3.4.5.

Question 3.5.5. Passe-t-il aujourd'hui et passera-t-il dans un mois ?

Partie 3 – Troisième problème de testabilité

Exercice 3.6. Problème de testabilité *(10 min)*

Considérons la méthode `ageMoisDateConsole()` qui prend la valeur de la date choisie en console.

Question 3.6.1. Essayez d'implémenter à nouveau le cas de test :

```
CT_age1 : ( « age entre le 28/2/24 et le 12/05/24 »,
            création d'un chien quelconque,
            naissance le (2024, 2, 28) et demande le (2024, 5, 12),
            2 mois)
```

Ce n'est tellement pas une bonne pratique, que les tests n'attendent pas qu'on écrive en console pour avancer et échouer :

https://intellij-support.jetbrains.com/hc/en-us/community/posts/115000556544-Why-can-t-I-input-anything-from-console-when-i-run-unit-test-with-JUNIT

Question 3.6.2. Quel est le problème de testabilité ?

Exercice 3.7. Injection de dépendance de classe *(20 min)*

Question 3.7.1. En tant que développeur, créer une classe DateConsole qui implémente l'interface DateProvider fournie et qui récupère une date en console (en y déplaçant les lignes de code qui font ce travail dans `ageMoisDateConsole()`.

Question 3.7.2. En tant que développeur ajouter un mécanisme d'injection de dépendance de classe qui va exploiter la classe DateConsole.

Question 3.7.3. Créez un stub de DateConsole qui renvoie toujours **(2024, 5, 12)**.

Question 3.7.4. Exploitez ce stub pour faire passer les tests.