TP 6 : Tests unitaires avec JUnit 5

Partie 1

Pour un contexte particulier, on a besoin de redéfinir la classe `ArrayList` pour bien gérer les occurrences d'un nombre. Pour cela, on utilise l'interface suivante :

```
package org.eclipse.main;

import java.util.List;

public interface Occurrence {
    Integer computeOccurrence(int i);
    Boolean containsNOccurrence(int i, int occ);
    void removeAllOccurrence(int i) throws ValueNotFoundException;
    void replaceAllOccurrence(int oldValue, int newValue) throws ValueNotFoundException;
    List<Integer> subListWithoutOccurrence(int i) throws ValueNotFoundException;
}
```

1. Créez l'interface `Occurrence`.
2. Créez la classe `OccurrenceList` qui étend de `ArrayList<Integer>` et implémente `Occurrence`.
3. Définissez un constructeur sans paramètre et un constructeur qui permet l'écriture suivante : `OccurrenceList list = new OccurrenceList(Arrays.asList(1, 2, 3, 2, 3, 3))`.
4. Implémentez toutes les méthodes abstraites de l'interface `Occurrence` :
   - `computeOccurrence(int i)` retourne le nombre d'occurrence de `i` dans la liste.
   - `containsNOccurrence(int i, int occ)` retourne `true` si la liste contient au moins `occ` occurrences de `i`, `false` sinon.
   - `removeAllOccurrence(int i)` supprime toutes les occurrences de `i` si ce dernier est dans la liste, sinon elle lève une exception de type `ValueNotFoundException` (une exception à définir).
   - `replaceAllOccurrence(int oldValue, int newValue)` remplace toutes les occurrences de `oldValue` par `newValue` si `oldValue` est dans la liste, sinon elle lève aussi une exception de type `ValueNotFoundException`.
   - `subListWithoutOccurrence(int i)` retourne une nouvelle sous-liste sans les occurrences de `i` (la liste originelle ne sera pas modifiée). Si la liste ne contient aucune occurrence de `i` elle lève une exception de type `ValueNotFoundException`.
5. En utilisant **JUnit 5**, créer une classe de test pour `OccurrenceList` ayant comme attribut : `OccurrenceList list = new OccurrenceList(Arrays.asList(1, 2, 3, 2, 3, 3))`.
6. Utilisez toutes les méthodes d'assertion, vues en cours, pour tester et valider les trois dernières méthodes de la classe `OccurrenceList`.
7. Utilisez `@RepeatedTest(3)` et `RepetitionInfo` pour tester `containsNOccurrence(int i, int occ)`.
8. Utilisez `@ParameterizedTest` et `@ValueSource(ints = { 1, 2, 3 })` pour tester `computeOccurrence(int i)`.

Partie 2

Considérons l'interface suivante :

```
package org.eclipse.main;

import java.util.List;

public interface Average {
    float compute(List<Integer> l);
}
```

1. Créez l'interface `Occurrence`.
2. Dans `OccurrenceList`, déclarez un attribut de type `Average`.
3. Définissez un constructeur qui permet l'écriture suivante :
   `OccurrenceList list = new OccurrenceList(average, Arrays.asList(1, 2, 3, 2, 3, 3))`.
4. Définissez une méthode `avgWithoutOccurrence(int i)` qui utilise la méthode `compute` de `Average` pour retourner la moyenne de la sous liste qui ne contient aucune occurrence de `i`.
5. Dans `OccurrenceListTest`, testez `avgWithoutOccurrence(int i)` en utilisant une classe anonyme héritant de `Average` (STUB).
6. Refaites le même test en remplaçant le STUB par une expression Lambda.
7. Définissez un mock sur `Average`.
8. Utilisez `when ... thenReturn` pour tester `avgWithoutOccurrence(int 2)`.
9. Vérifiez que le mock a été utilisé.